Lecture IV:
Convolutional Neural Network (CNN)

Three Steps for Deep Learning

Step 1: Neural Network → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……

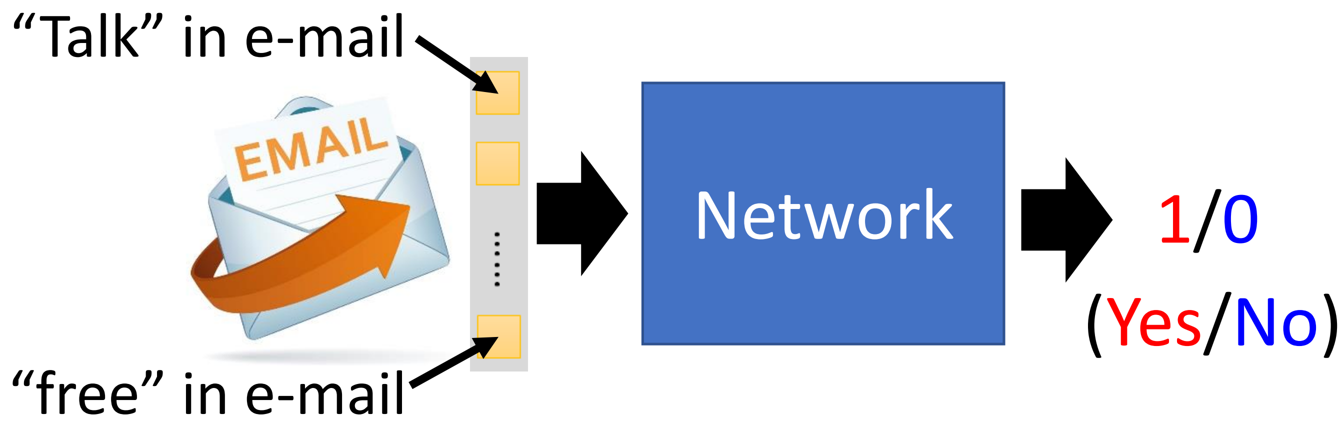
Now If you want to find a function

If you have lots of function input/output (?) as training data

→ You can use deep learning

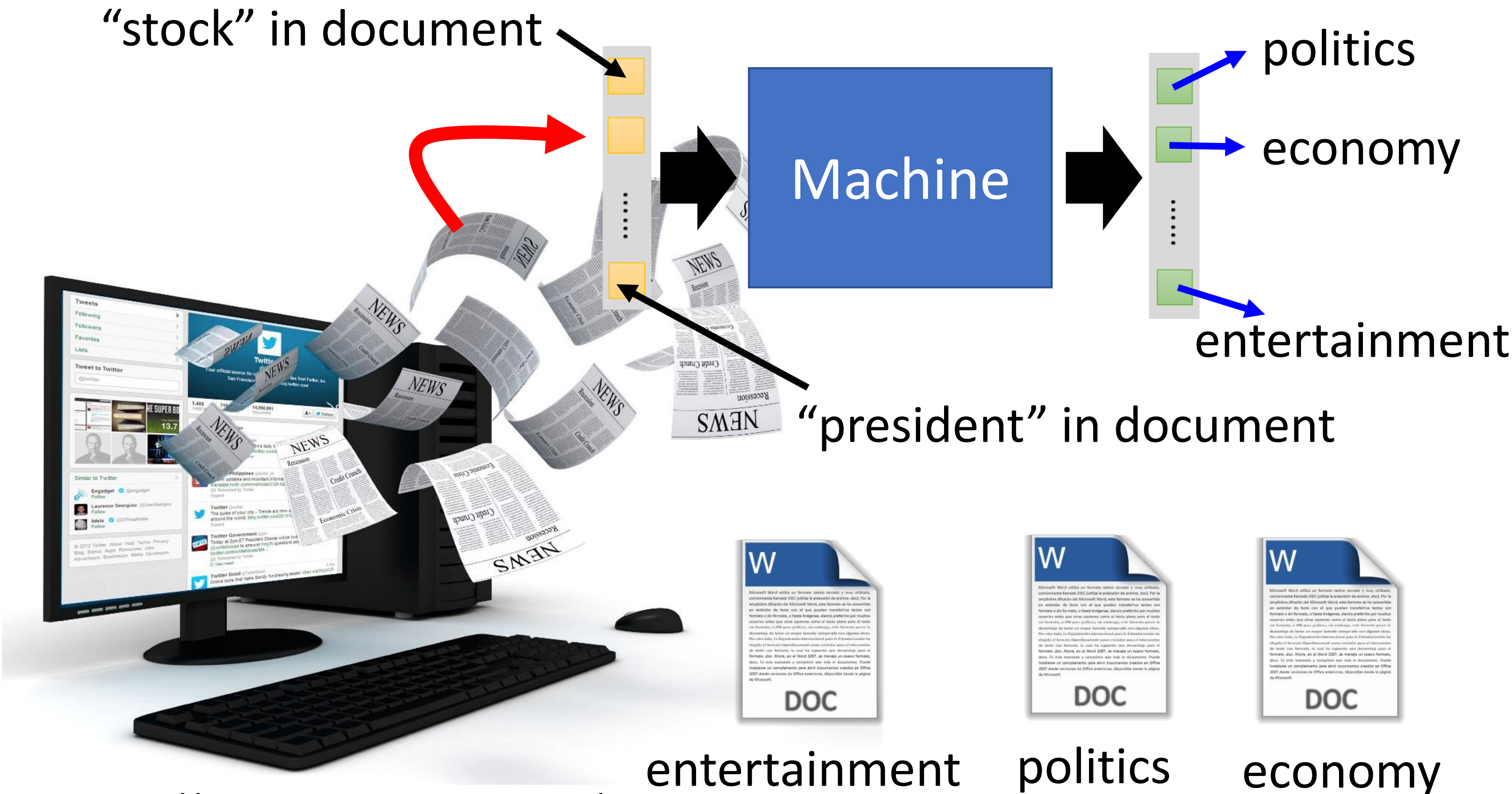
For example, you can do …….

Spam filtering



(http://spam-filter-review.toptenreviews.com/)

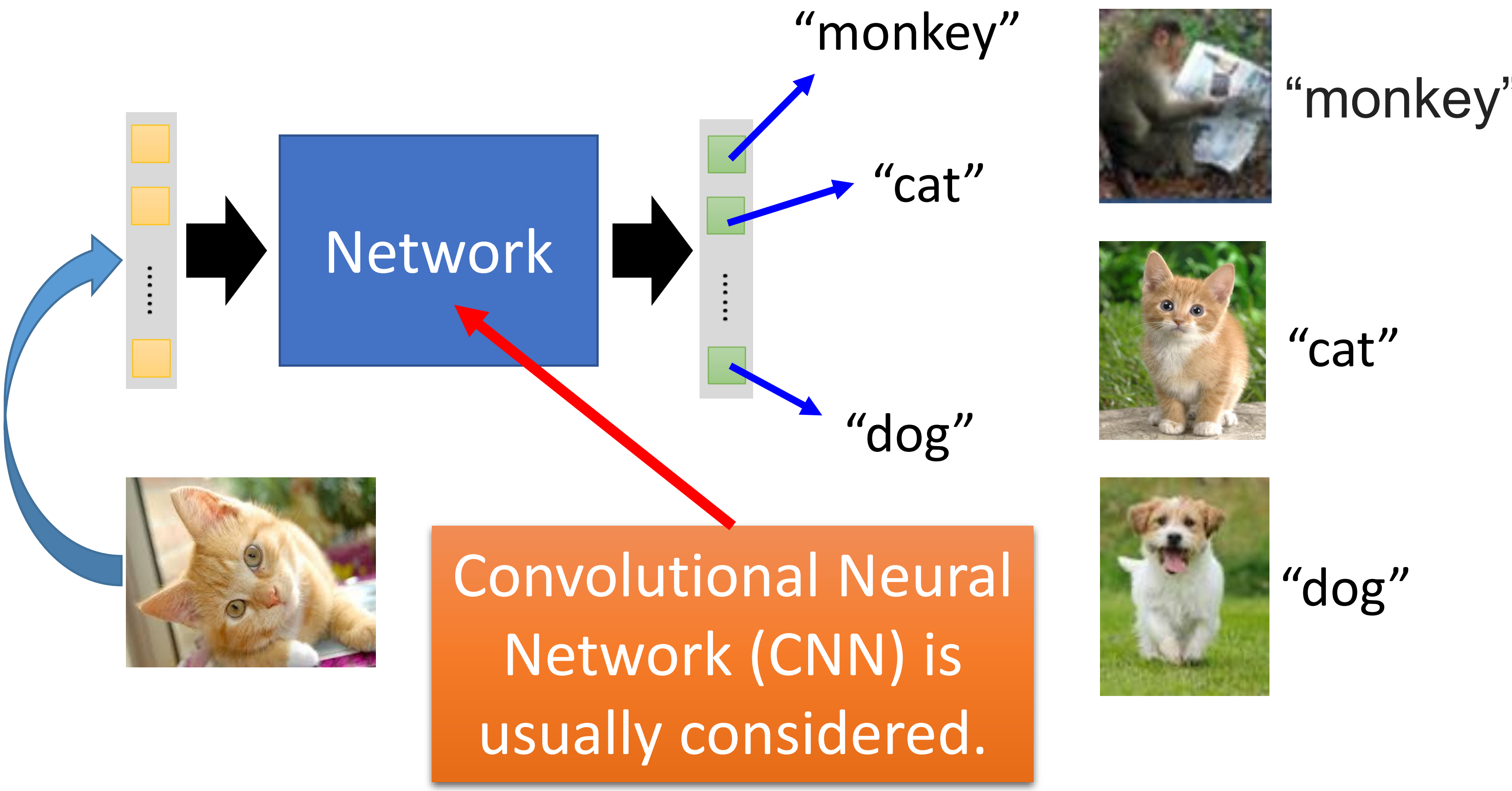
For example, you can do …….



http://top-breaking-news.com/

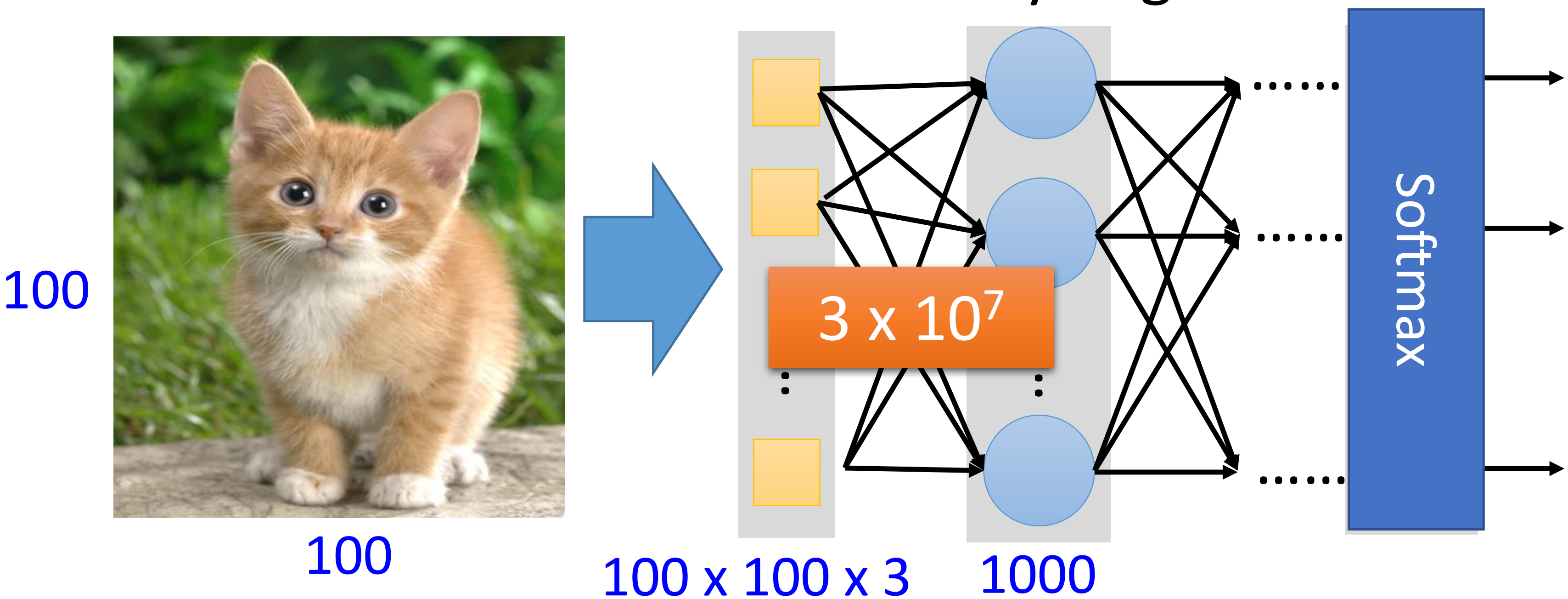
For example, you can do …….

- Image Recognition



Why CNN for Image?

- When processing image, the first layer of fully connected network would be very large



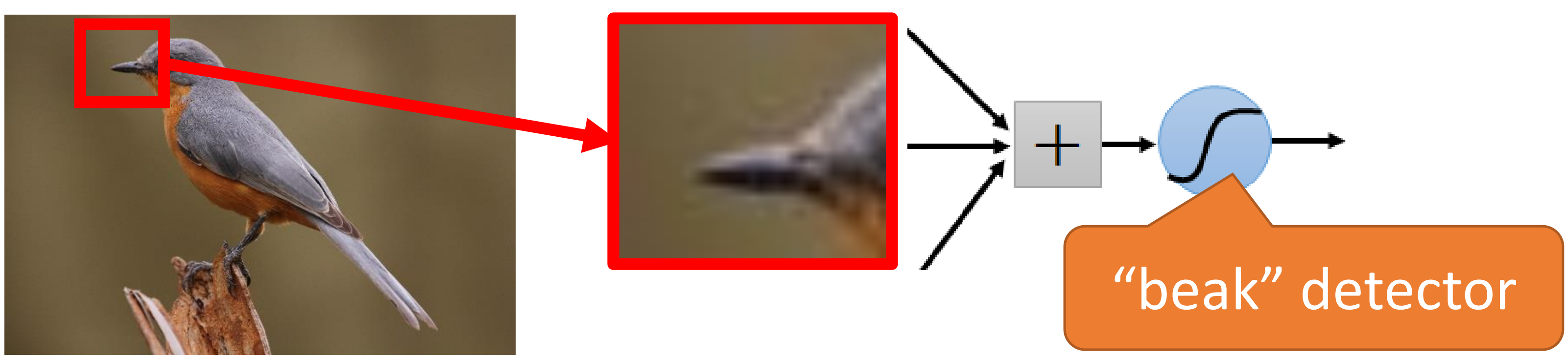
Can the fully connected network be simplified by considering the properties of image processing?

Why CNN for Image

- Some patterns are much smaller than the whole image

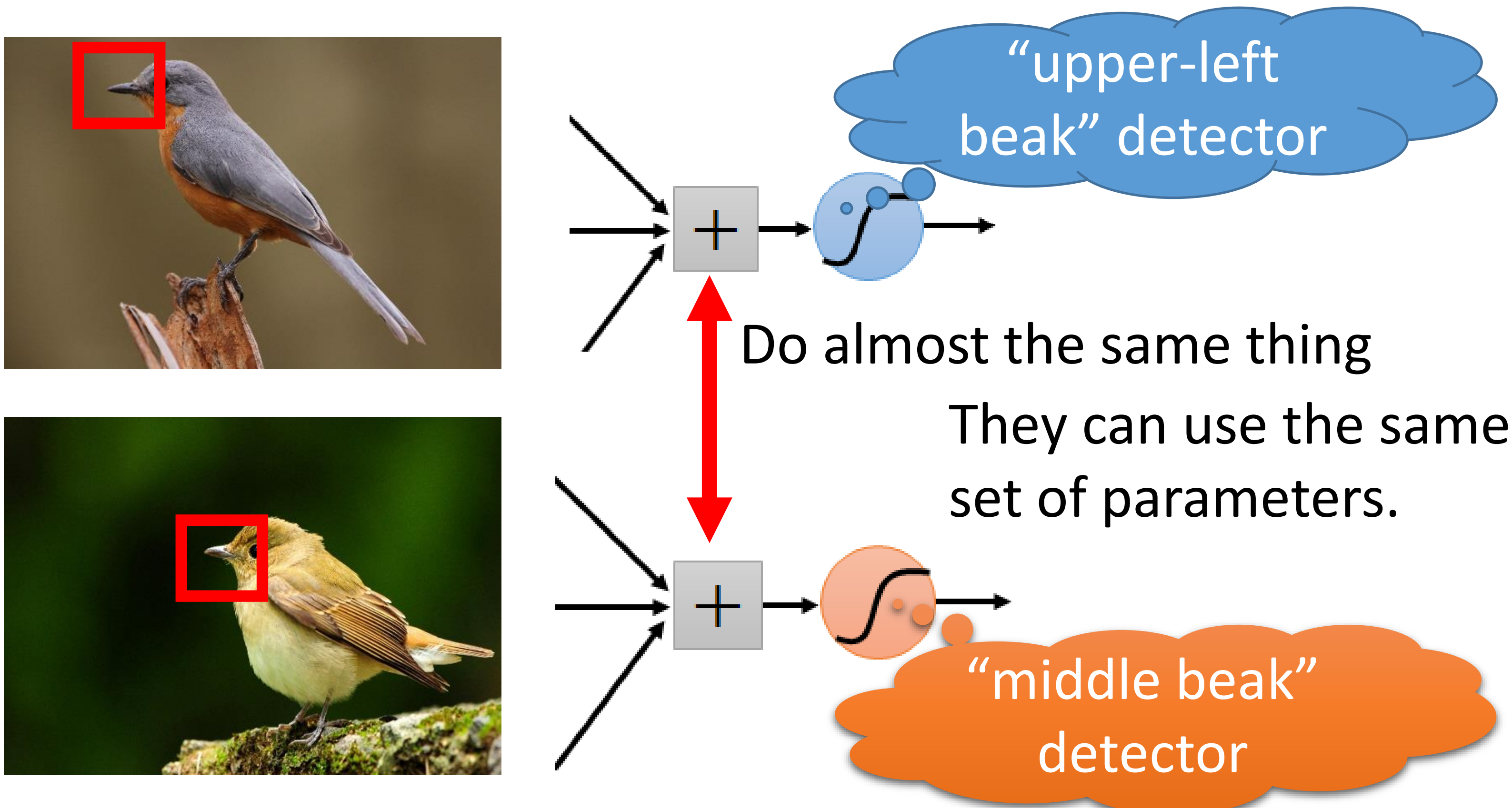
A neuron does not have to see the whole image to discover the pattern.

Connecting to small region with less parameters



Why CNN for Image

- The same patterns appear in different regions.



Why CNN for Image

- Subsampling the pixels will not change the object

bird

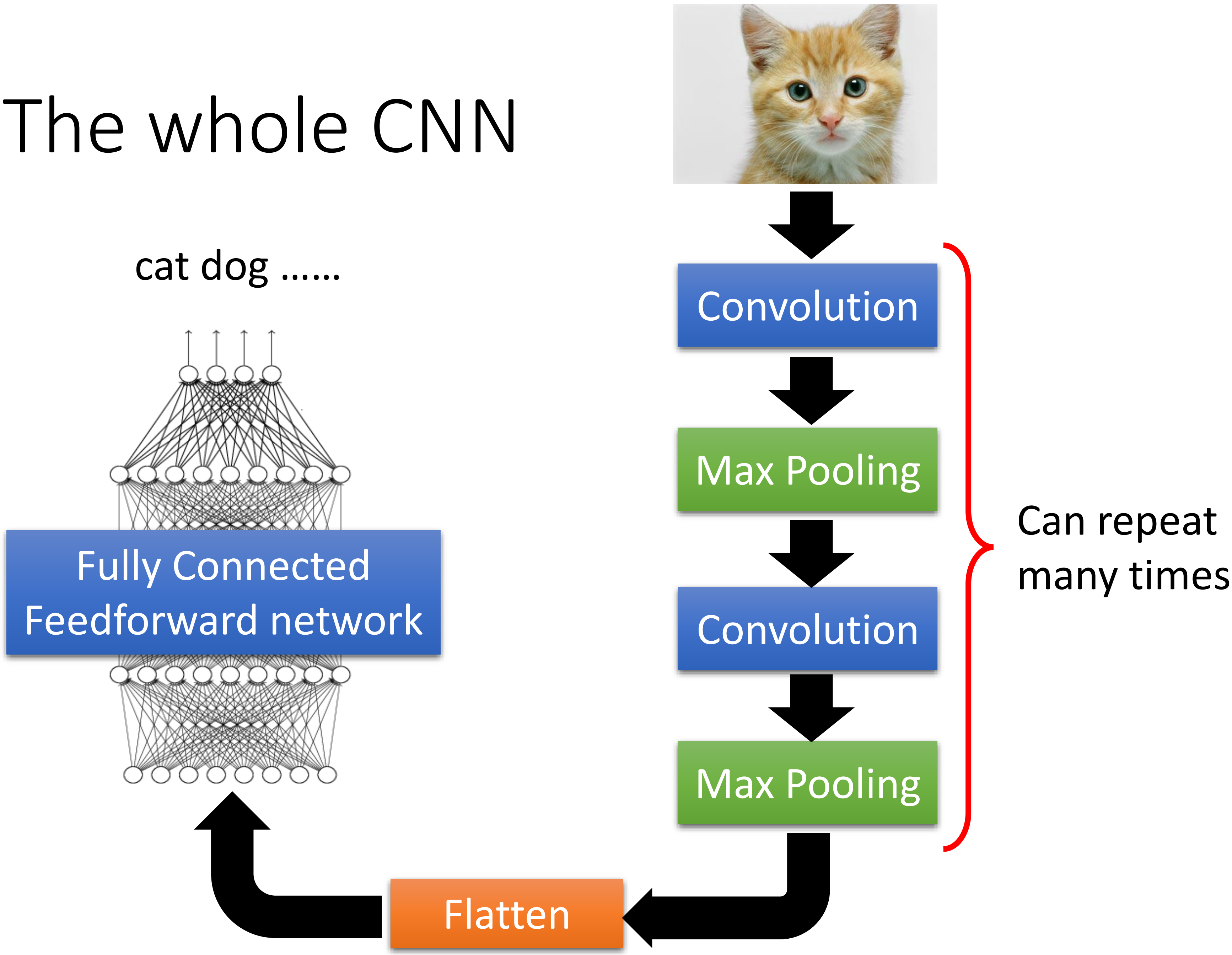
subsampling

bird

We can subsample the pixels to make image smaller

Less parameters for the network to process the image

The whole CNN



The whole CNN

Property 1

- ➢ Some patterns are much smaller than the whole image

Property 2

- ➢ The same patterns appear in different regions.

Property 3

- ➢ Subsampling the pixels will not change the object

Convolution

Max Pooling

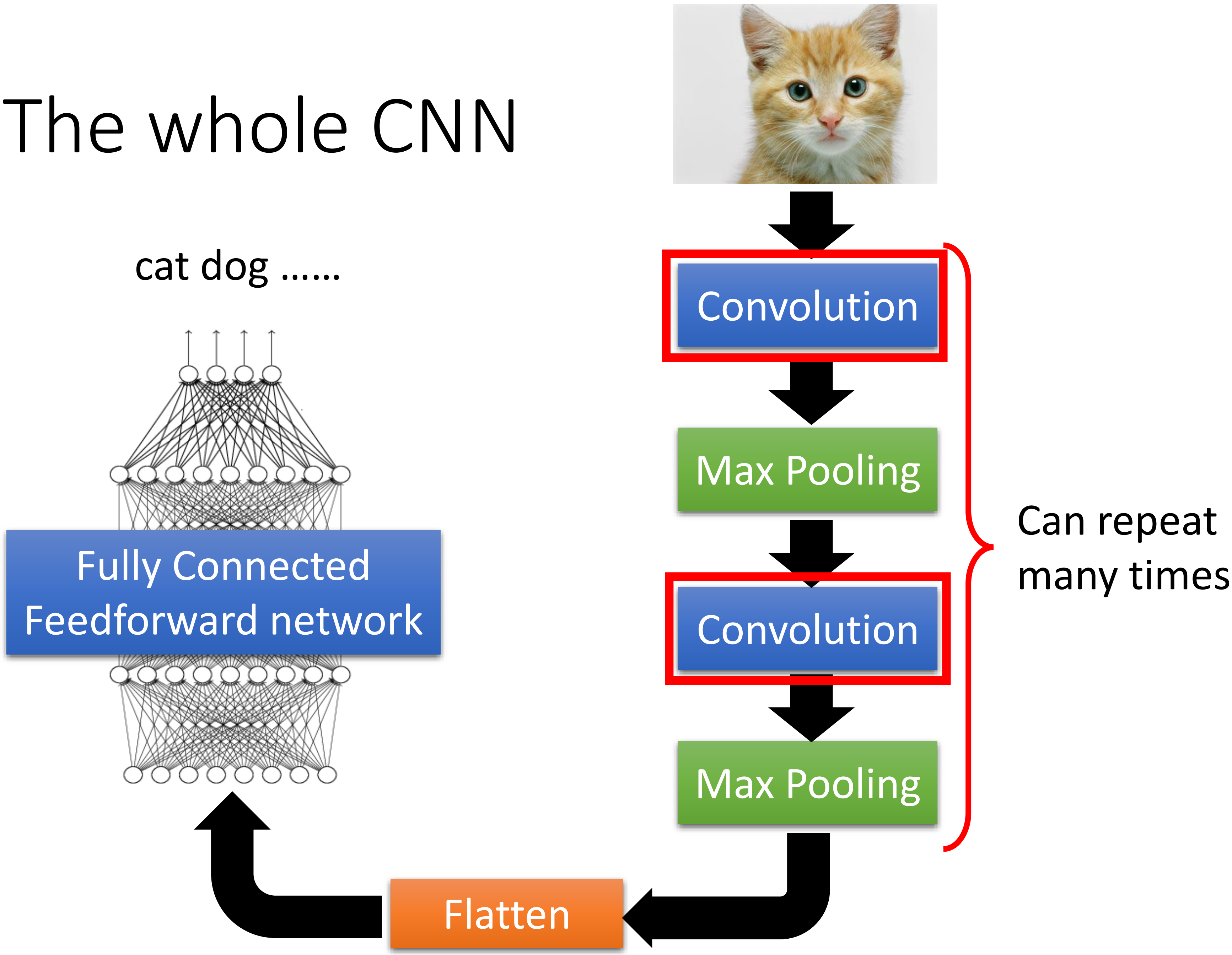
Convolution

Max Pooling

Can repeat many times

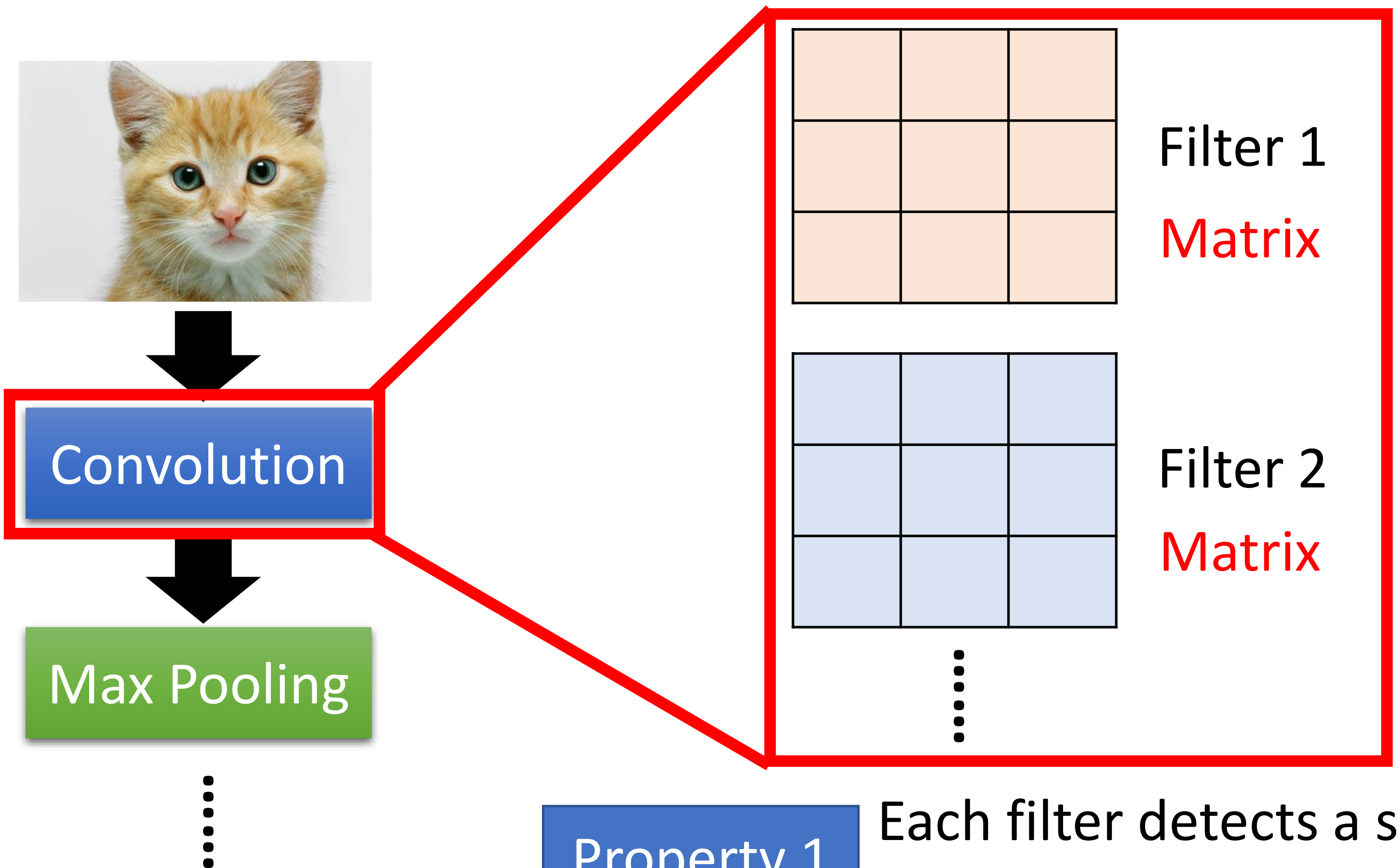
Flatten

The whole CNN



CNN – Convolution

The values in the matrices are learned from training data.



Property 1

Each filter detects a small pattern (3 x 3).

CNN – Convolution

The values in the matrices are learned from training data.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

Matrix

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

Matrix

⋮

Property 1

Each filter detects a small pattern (3 x 3).

CNN – Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

Property 2

CNN – Convolution

-1	1	-1
-1	1	-1
-1	1	-1

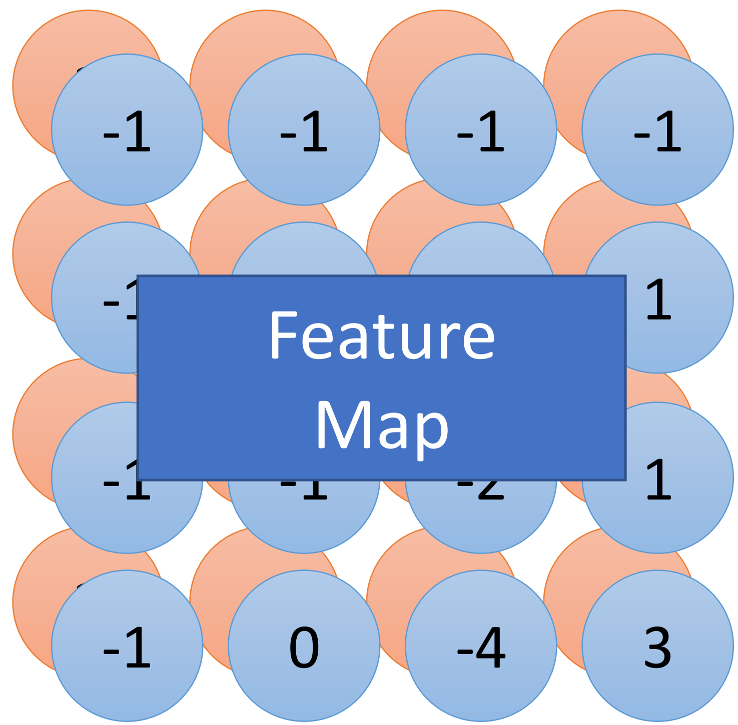
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Do the same process for every filter



4 x 4 image

CNN – Colorful image

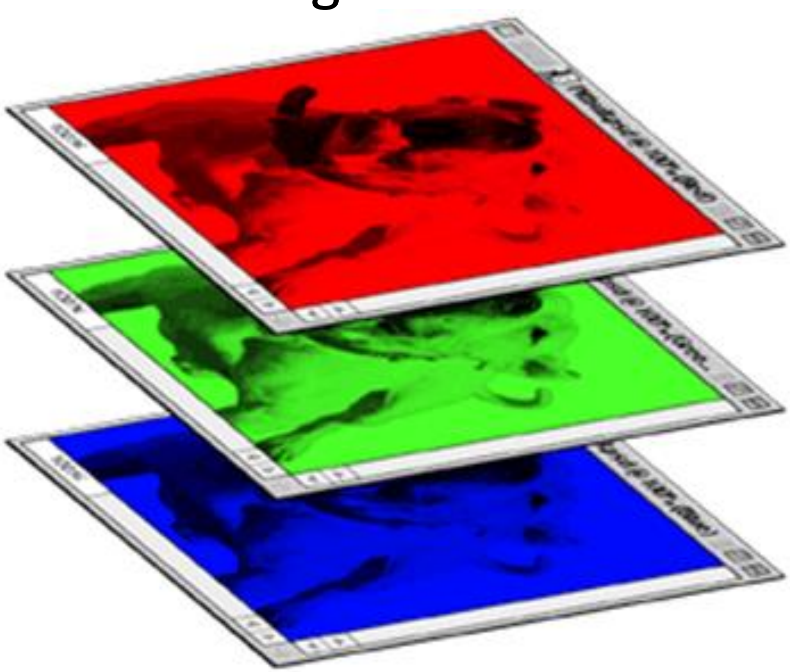
Filter 1

1	-1	-1
-1	1	-1
-1	-1	1

Filter 2

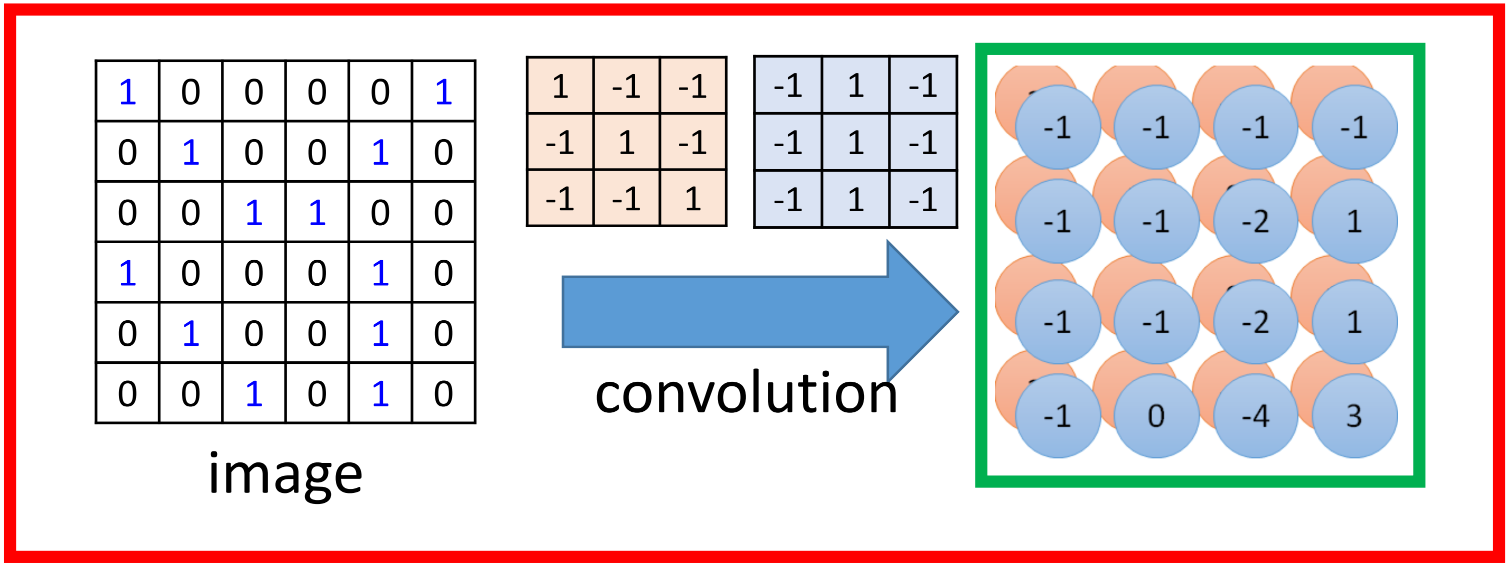
-1	1	-1
-1	1	-1
-1	1	-1

Colorful image

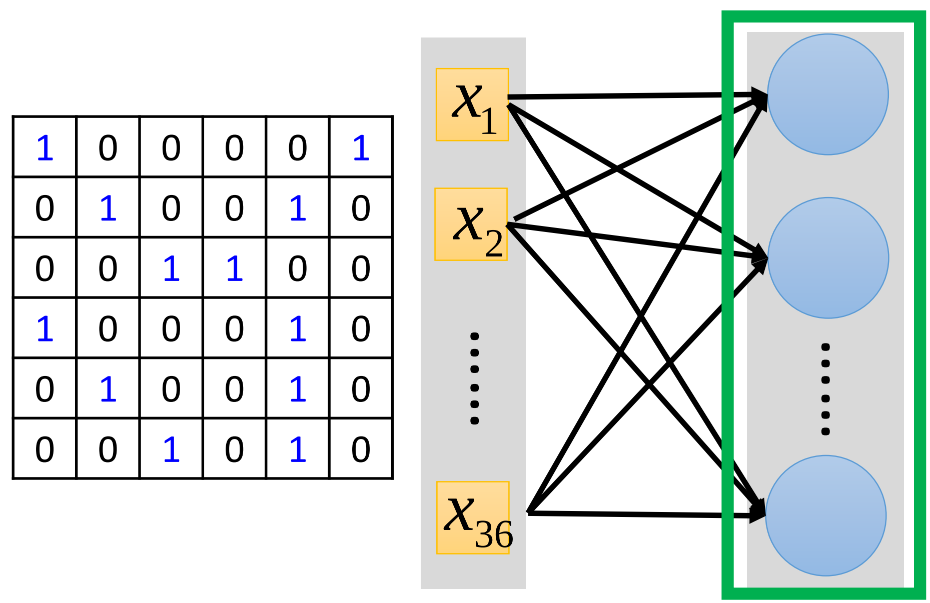


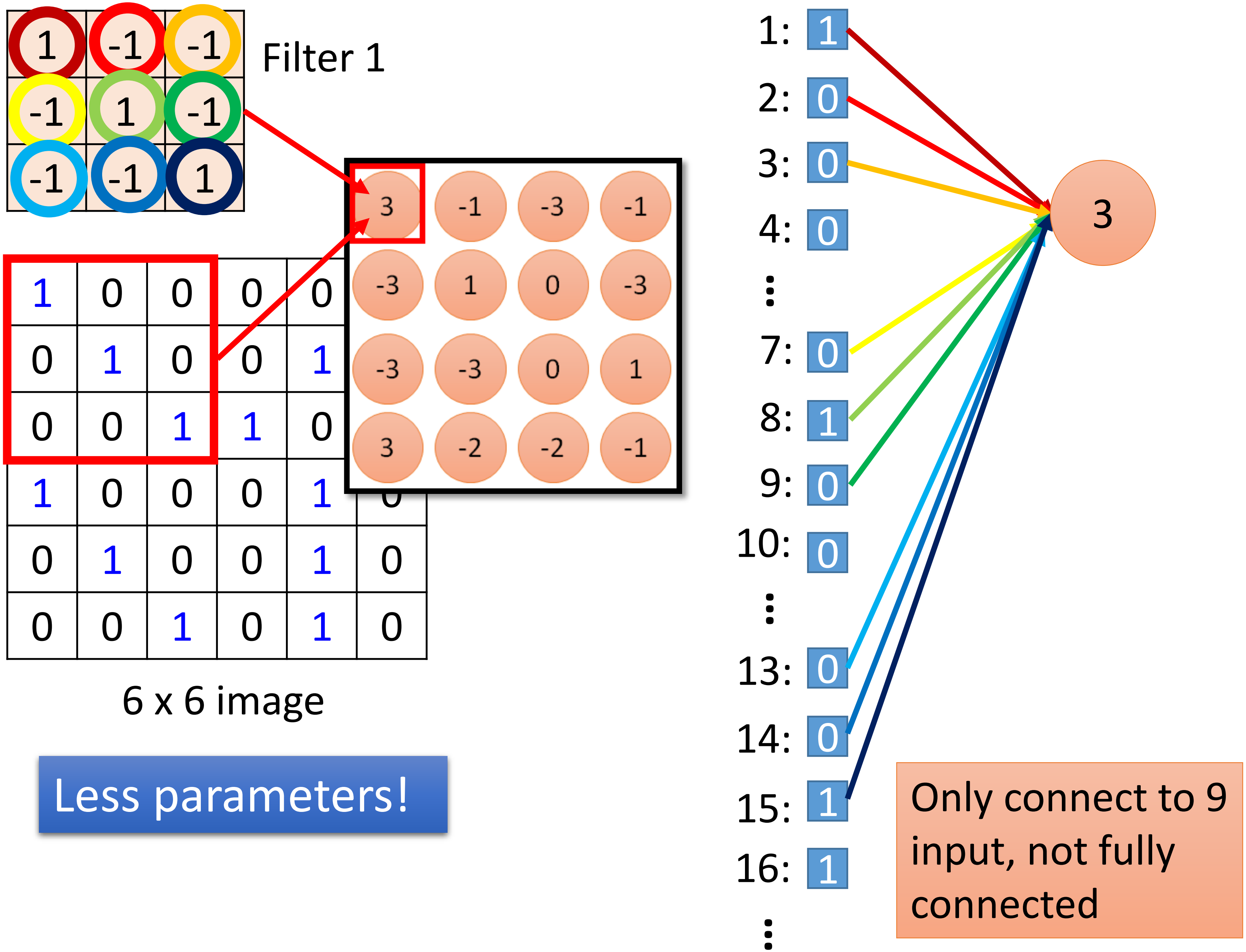
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Convolution v.s. Fully Connected

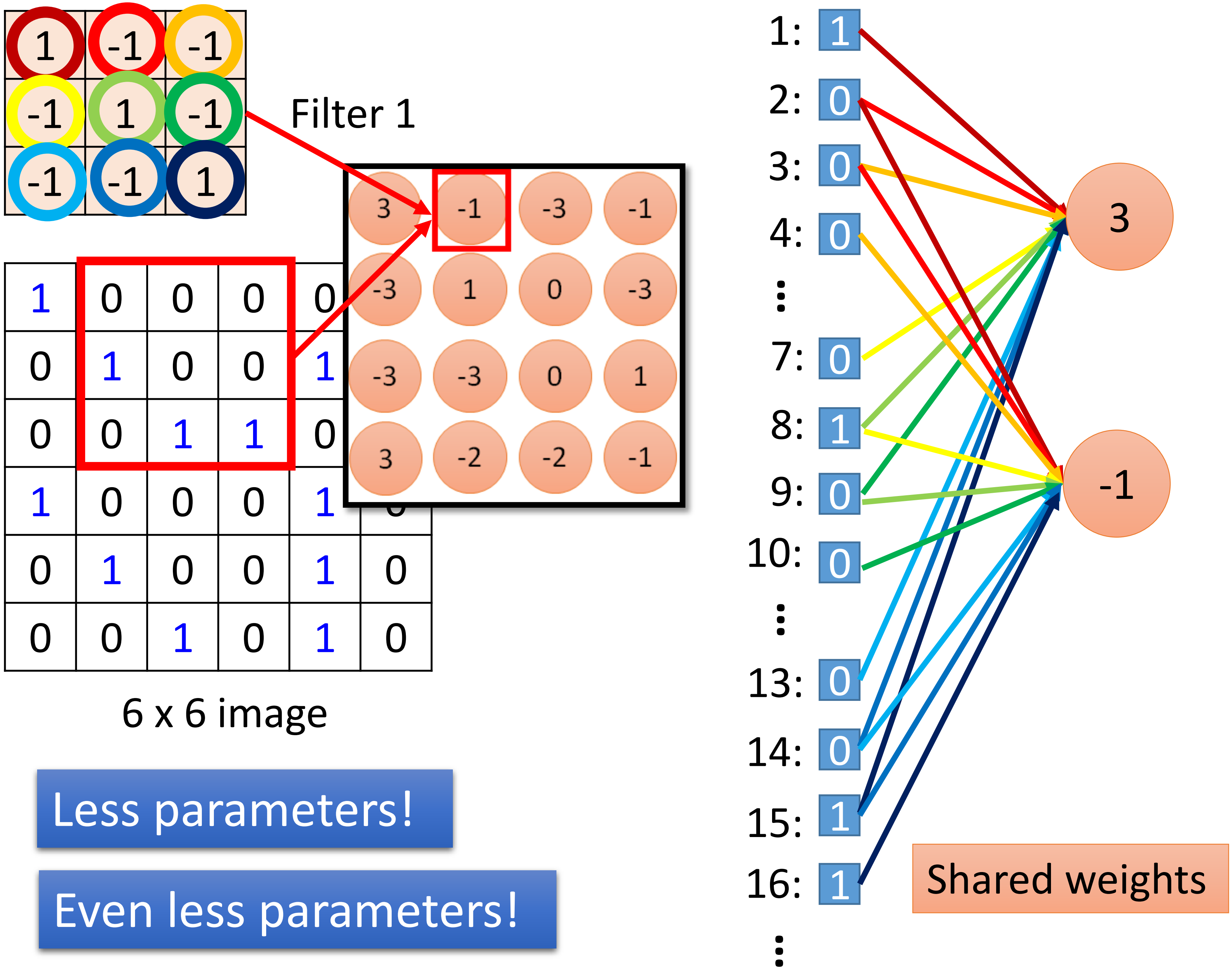


Fully-connected



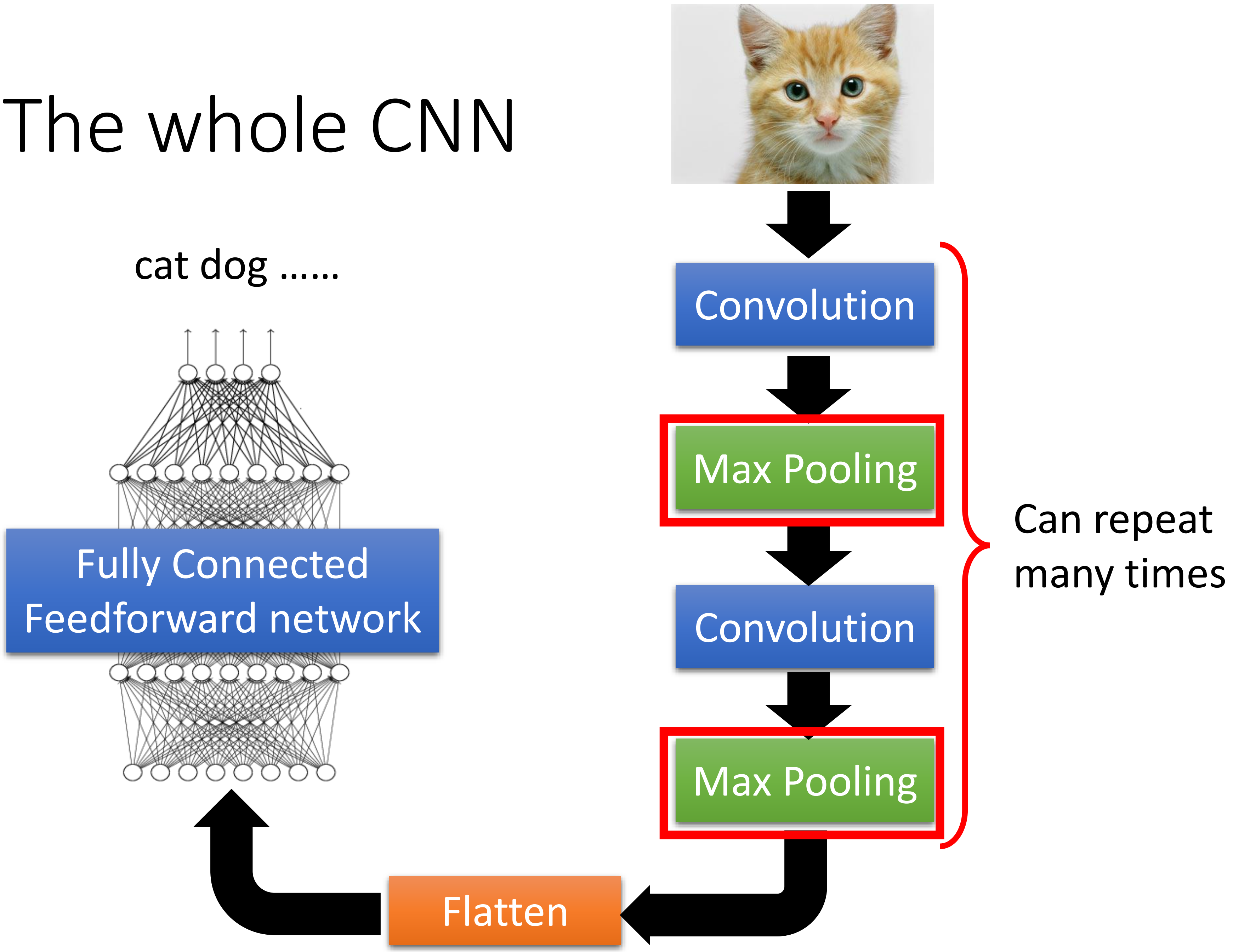


Filter 1
1 -1 -1
-1 1 -1
-1 -1 1
1 0 0 0 0
0 1 0 0 1
0 0 1 1 0
1 0 0 0 1
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
Less parameters!
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
Only connect to 9 input, not fully connected



Filter 1
6 x 6 image
Less parameters!
Even less parameters!
Shared weights
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
-1

The whole CNN



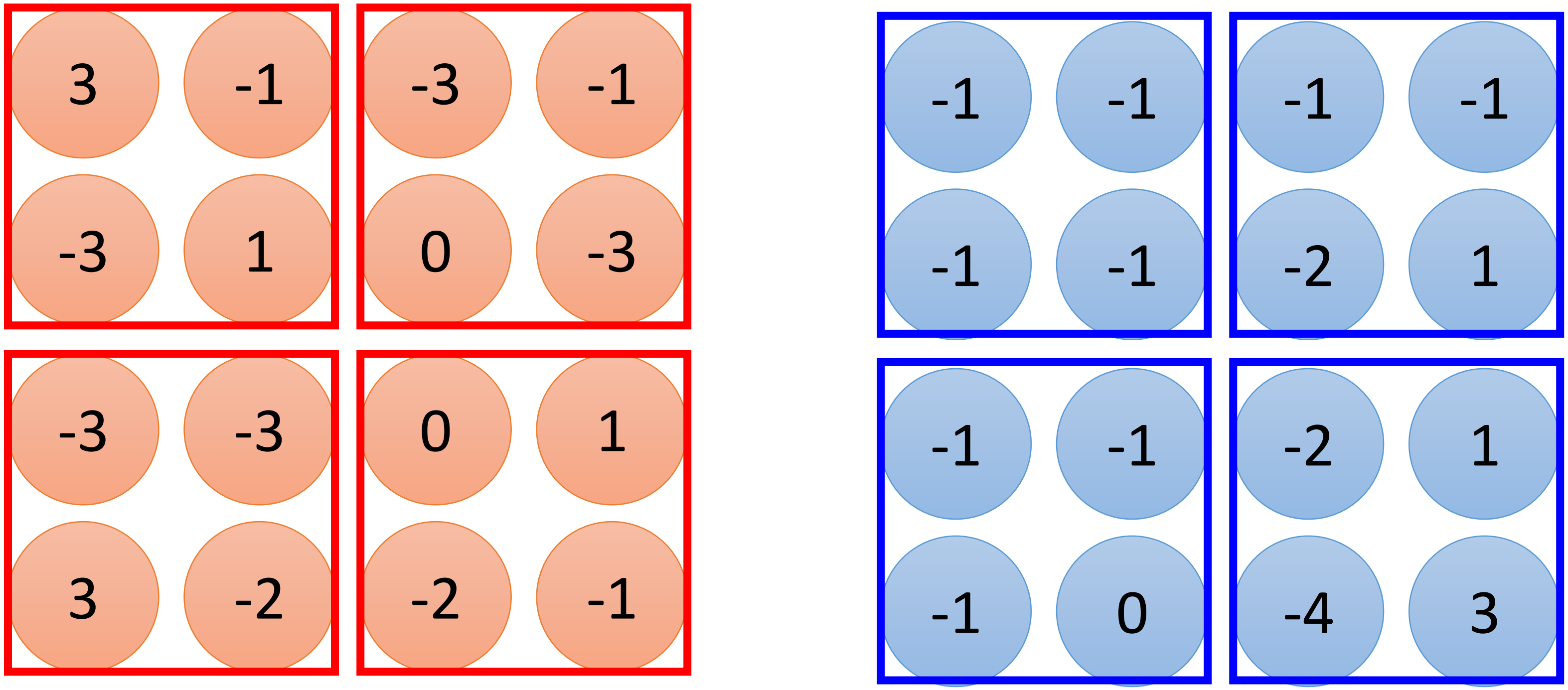
CNN – Max Pooling

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

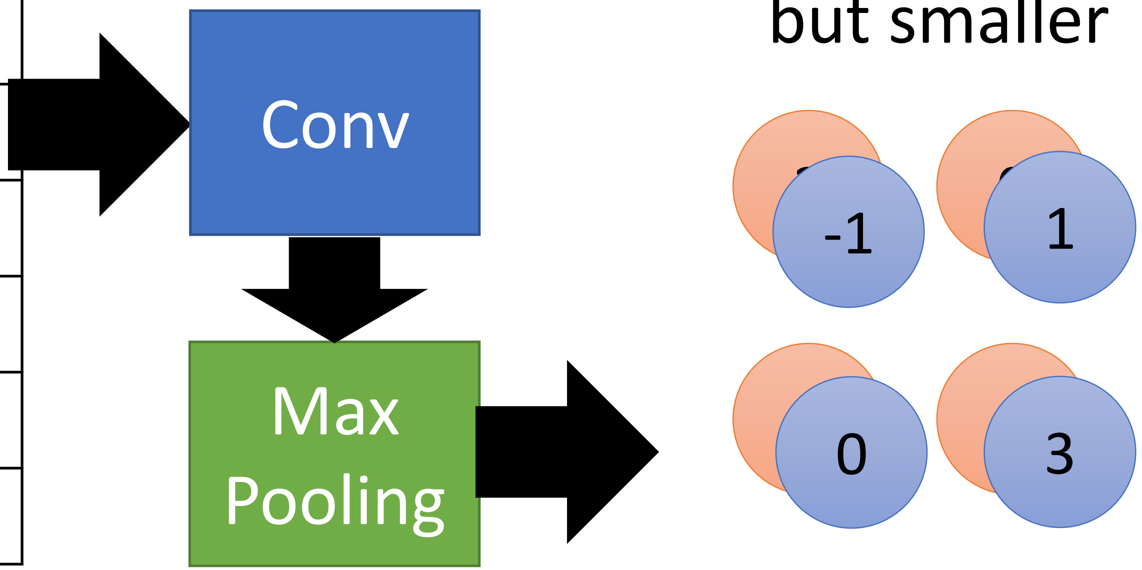


CNN – Max Pooling

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

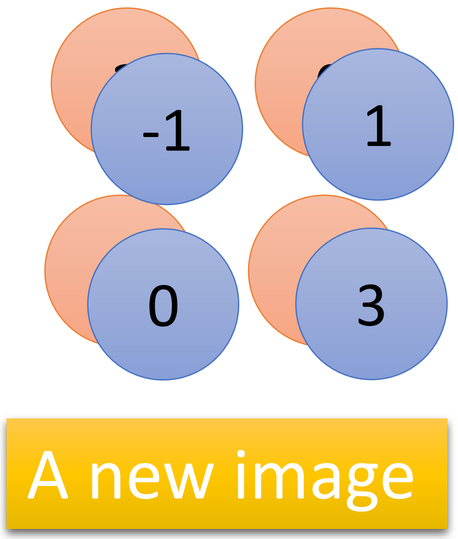
New image but smaller

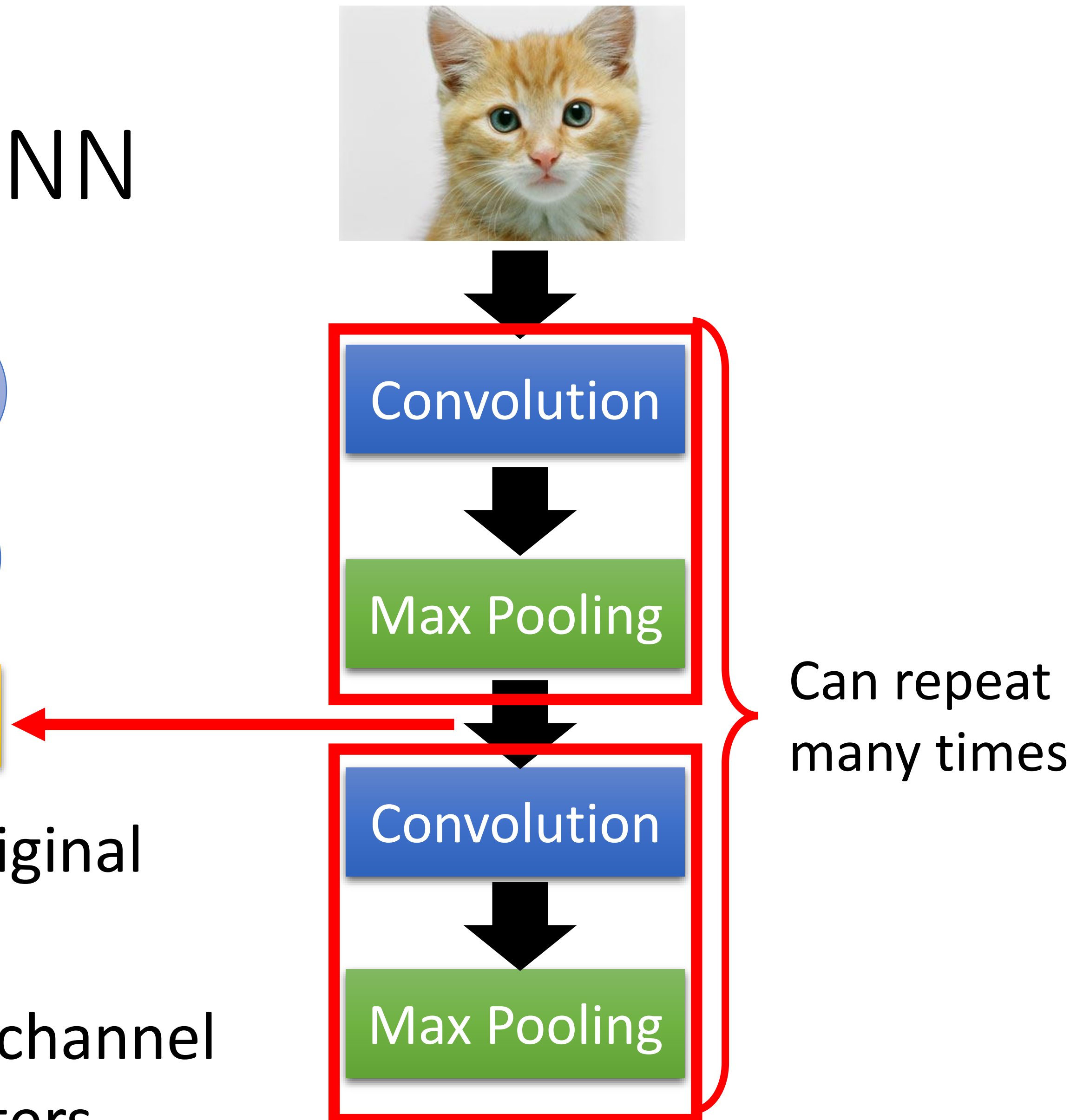


2 x 2 image

Each filter is a channel

The whole CNN

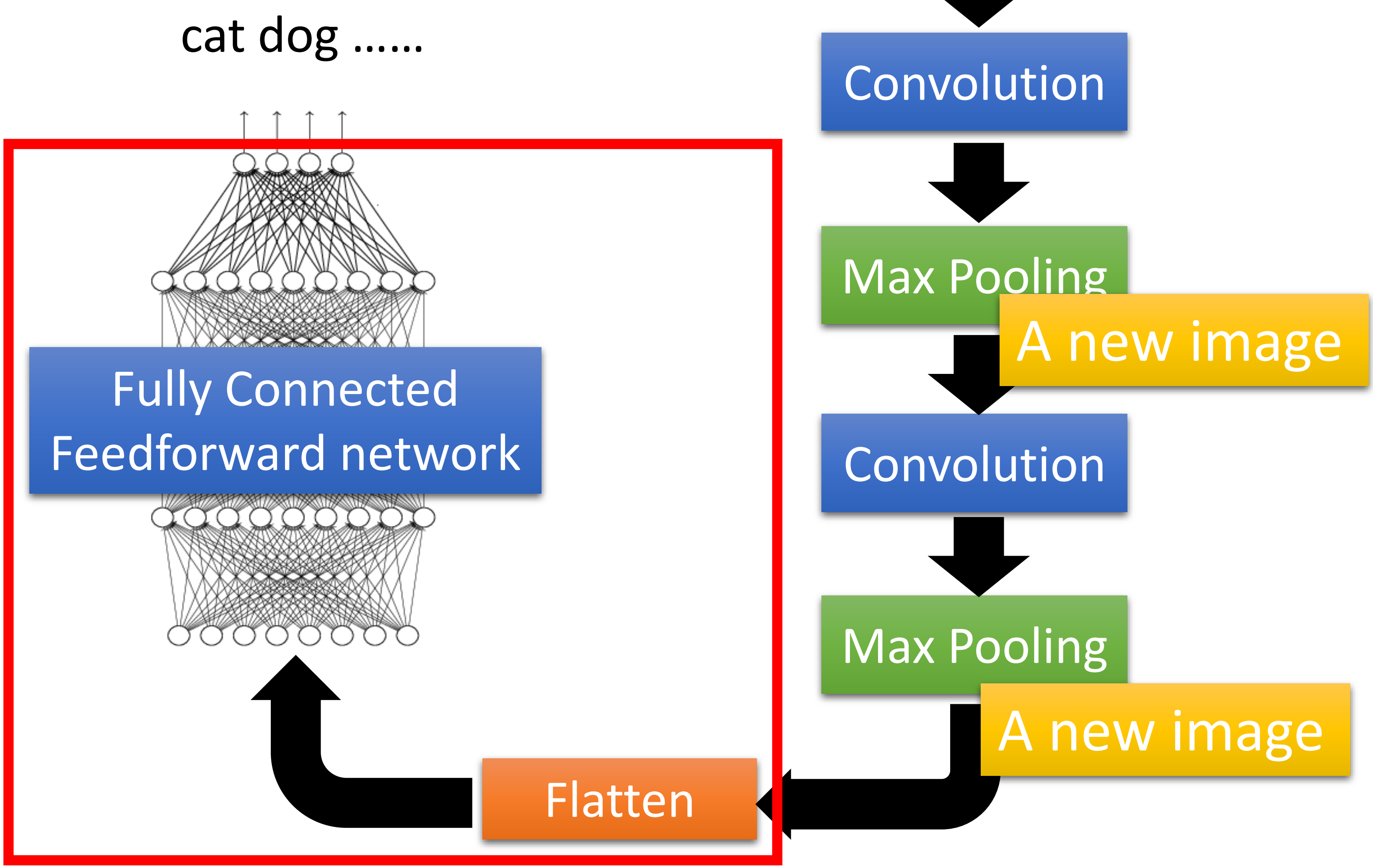




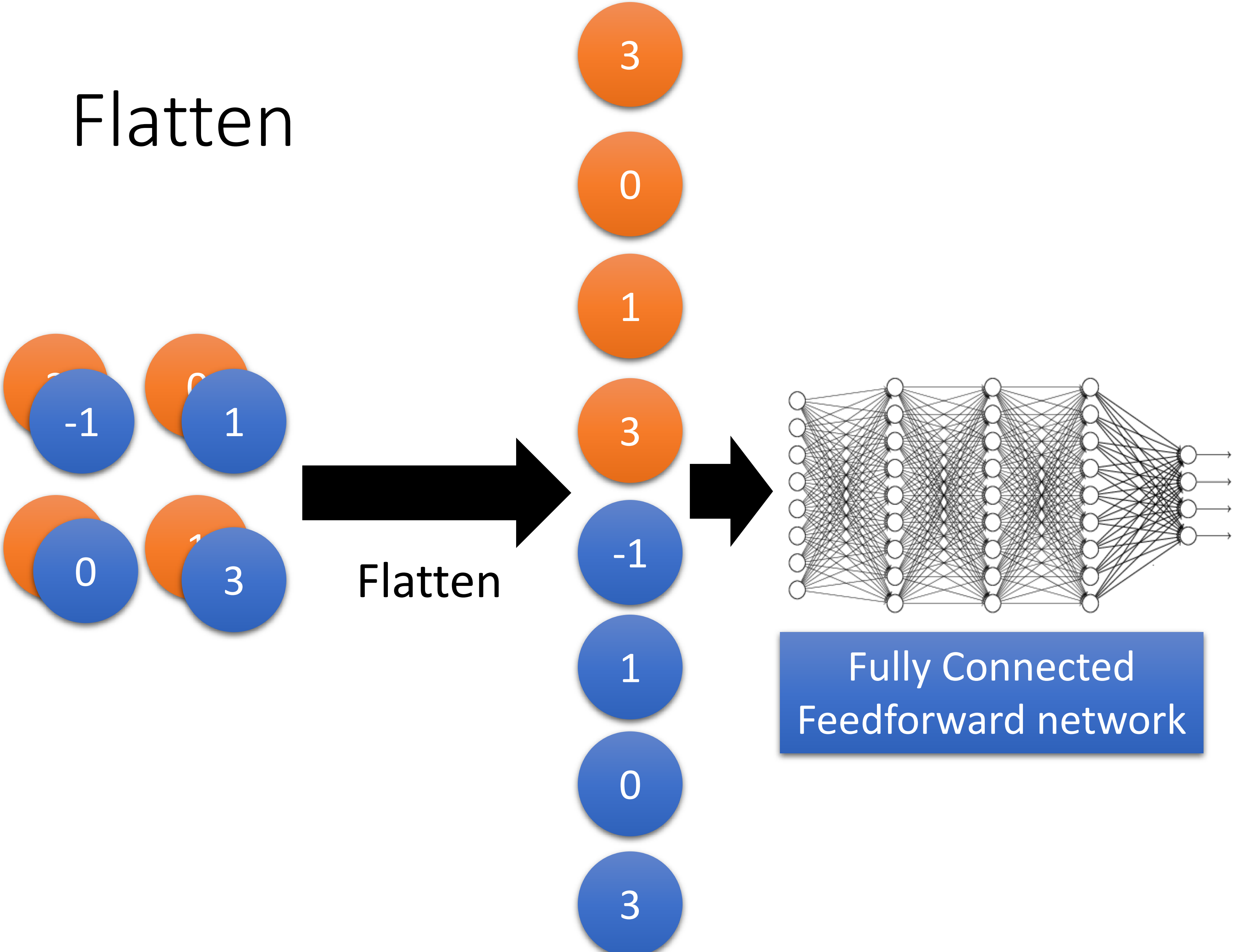
Smaller than the original image

The number of the channel is the number of filters

The whole CNN

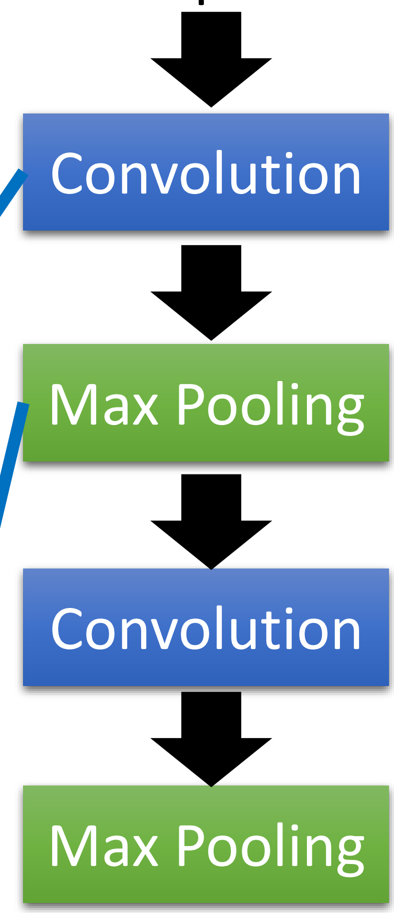


Flatten

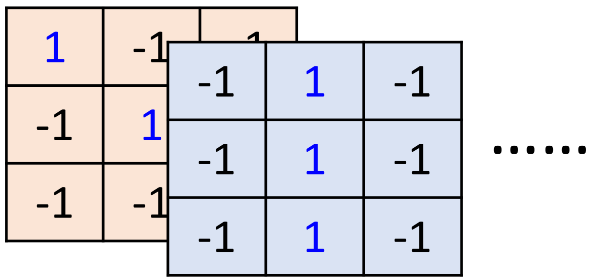

-1
1
0
3
Flatten
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***




```
model2.add( Convolution2D( 25,3,3,
            input_shape=(1,28,28) ) )
```

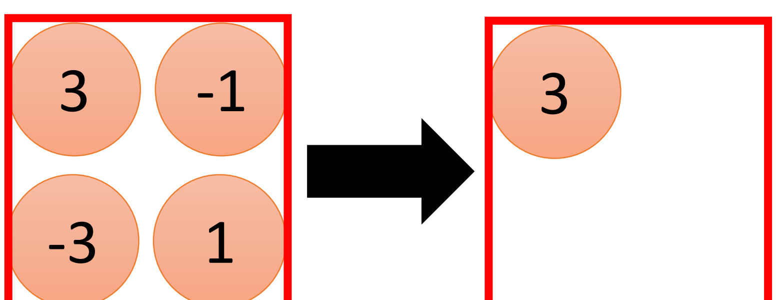


There are 25 3x3 filters.

Input_shape = (1 , 28 , 28)

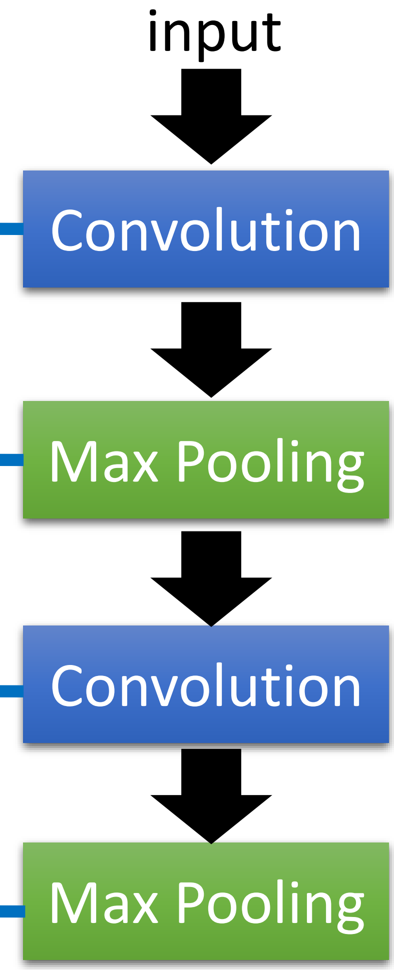
1: black/weight, 3: RGB   28 x 28 pixels

```
model2.add(MaxPooling2D((2,2)))
```



CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***



1 x 28 x 28

```
model2.add( Convolution2D( 25,3,3,
            input_shape=(1,28,28) ) )
```

How many parameters for each filter? 9

25 x 26 x 26

```
model2.add(MaxPooling2D((2,2)))
```

25 x 13 x 13

```
model2.add(Convolution2D(50,3,3))
```

How many parameters for each filter? 225

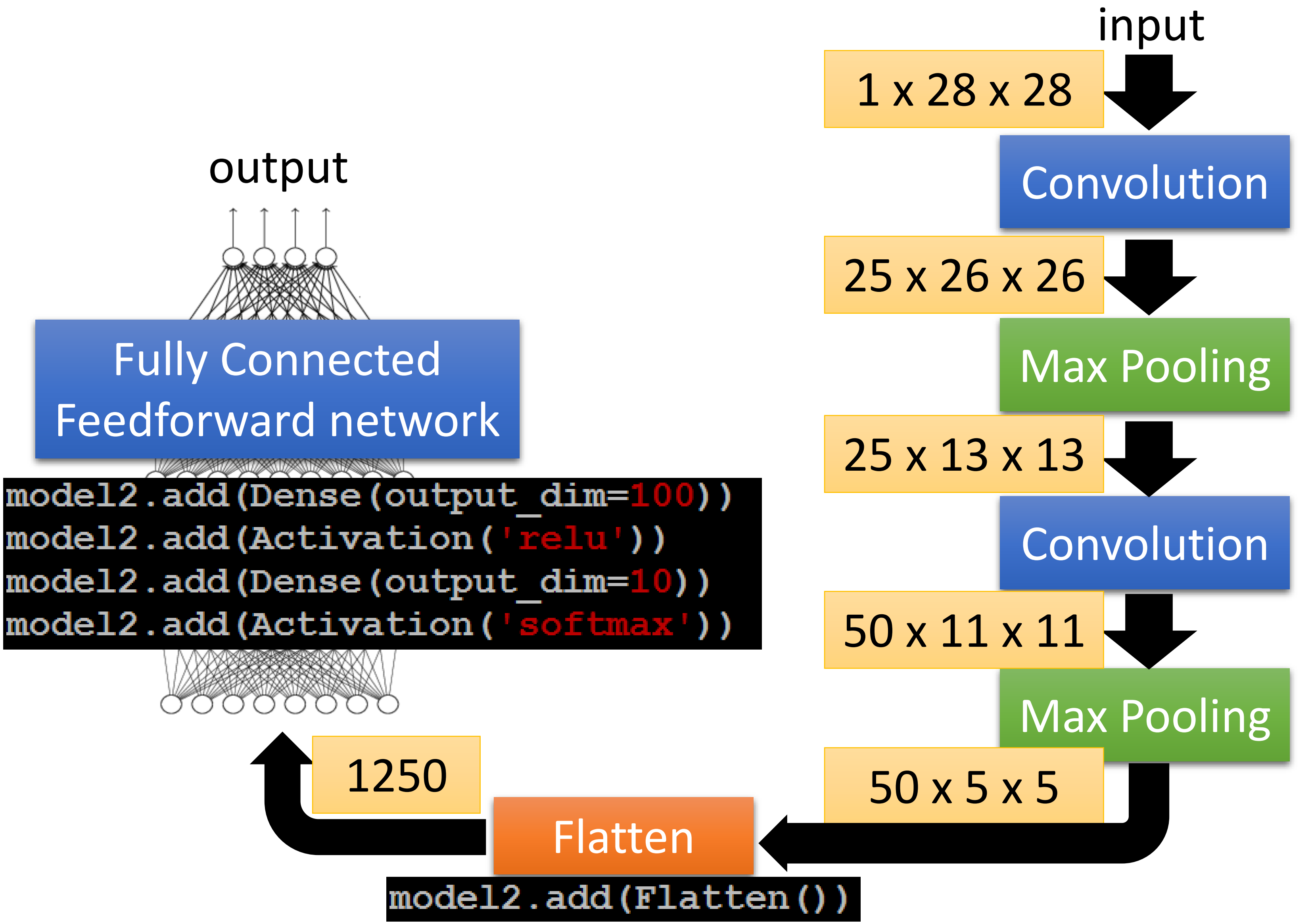
50 x 11 x 11

```
model2.add(MaxPooling2D((2,2)))
```

50 x 5 x 5

CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***



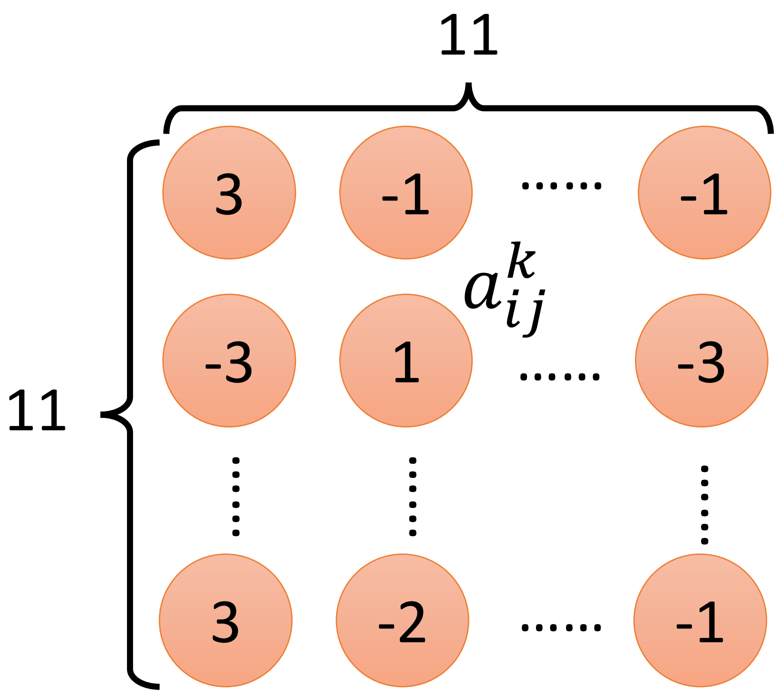
Live Demo

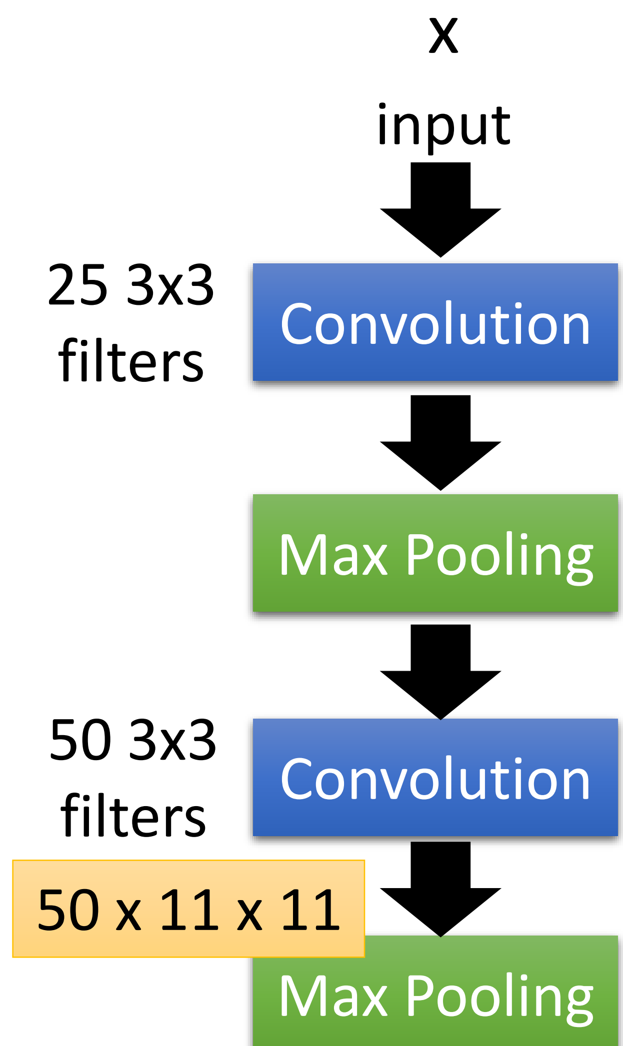
What does CNN learn?

The output of the k-th filter is a 11 x 11 matrix.

Degree of the activation of the k-th filter: $a^k = \sum_{i=1}^{11} \sum_{j=1}^{11} a_{ij}^k$

$x^* = arg \max_x a^k$ (gradient ascent)



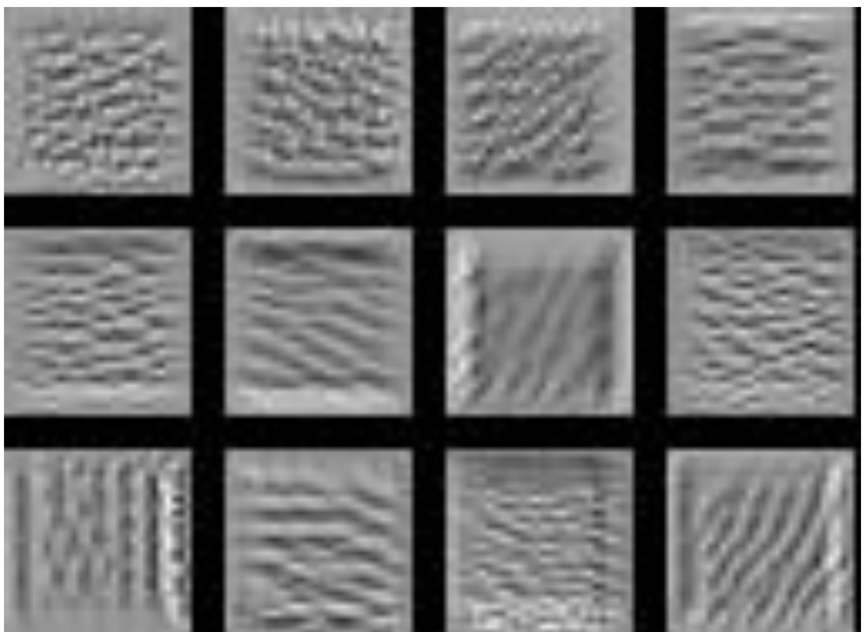


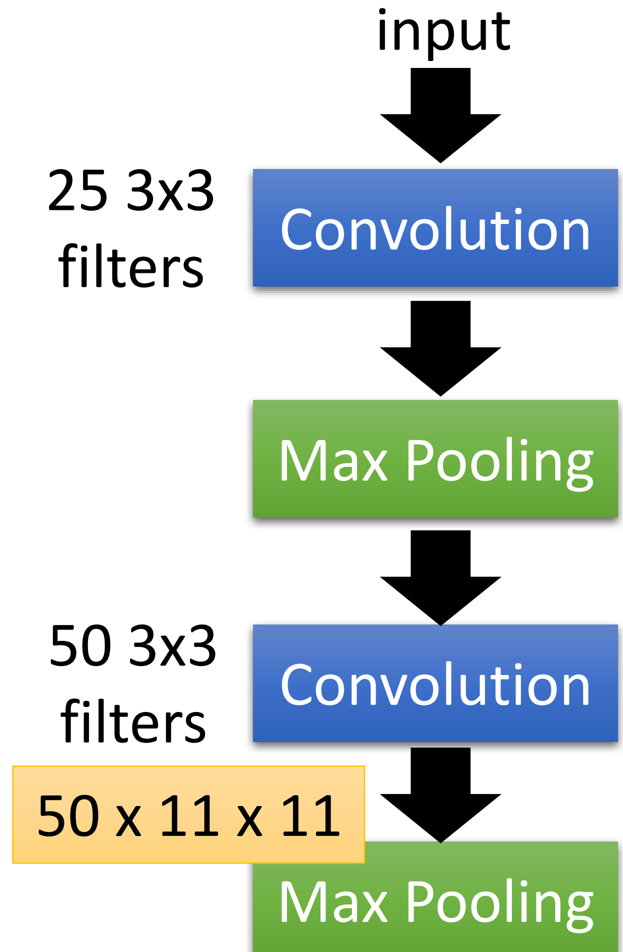
What does CNN learn?

The output of the k-th filter is a 11 x 11 matrix.

Degree of the activation of the k-th filter: $a^k = \sum_{i=1}^{11} \sum_{j=1}^{11} a_{ij}^k$

$x^* = arg\max_{x} a^k$ (gradient ascent)

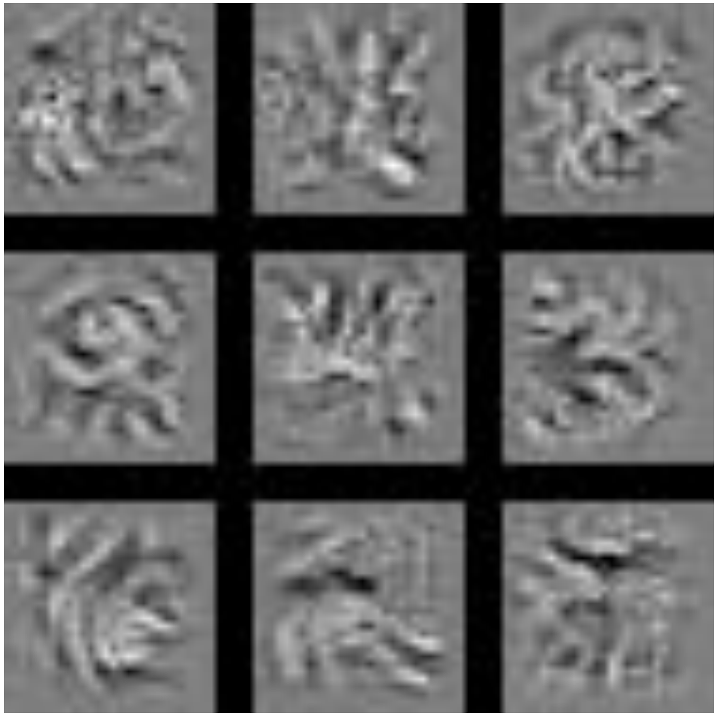




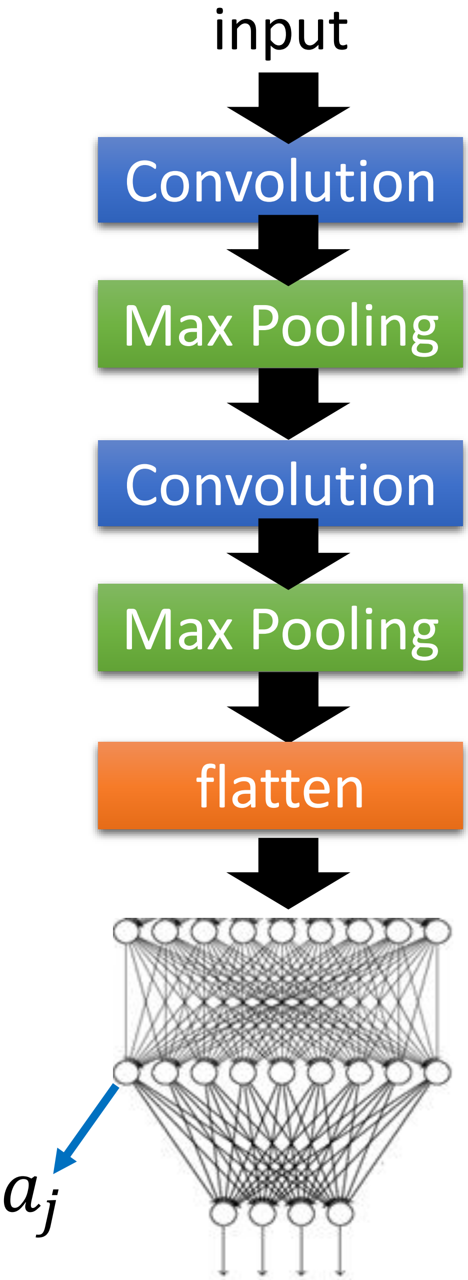
Each small figure corresponds to a filter.

What does CNN learn?

Find an image maximizing the output of neuron: $$x^* = arg\max_x a^j$$



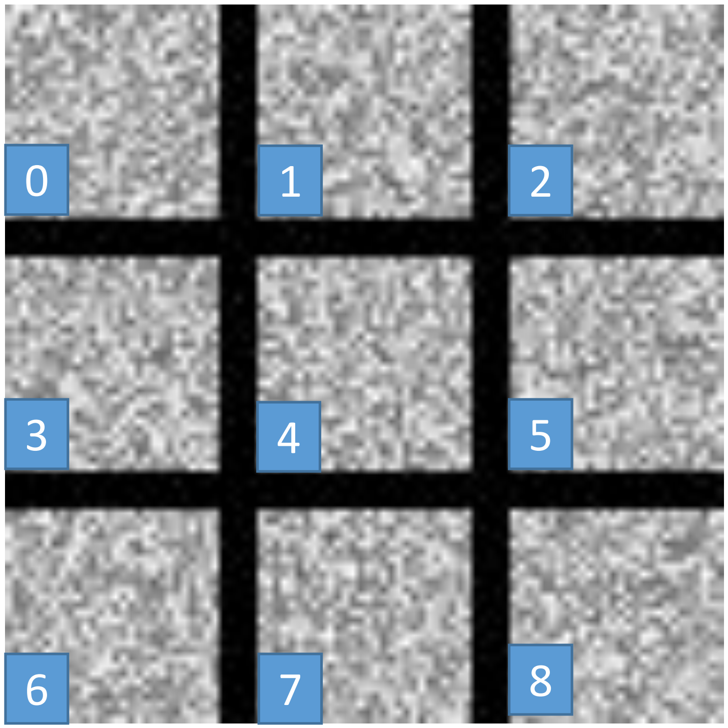
Each figure corresponds to a neuron

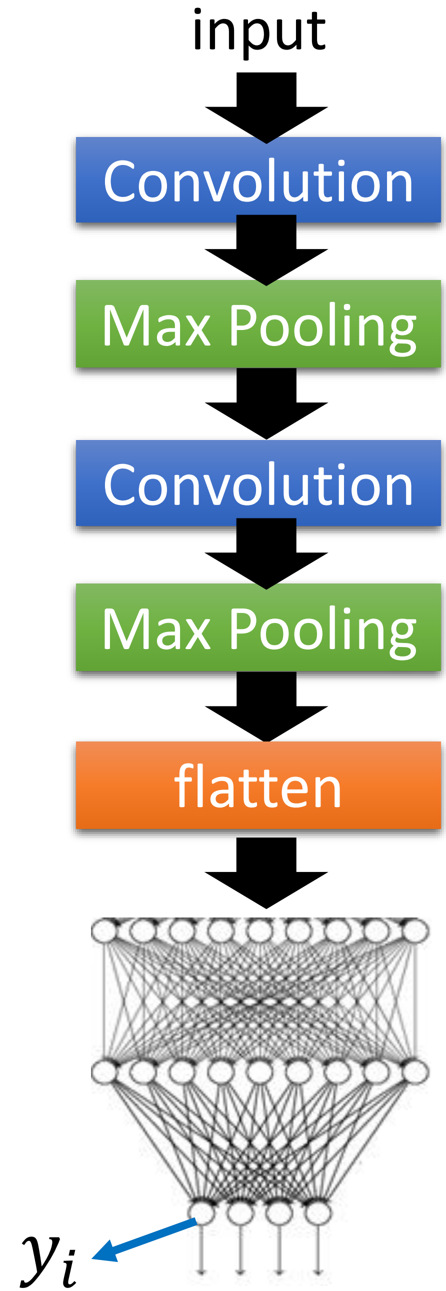


What does CNN learn?

$$x^* = arg \max_x y^i$$

Can we see digits?





Deep Neural Networks are Easily Fooled

https://www.youtube.com/watch?v=M2IebCN9Ht4

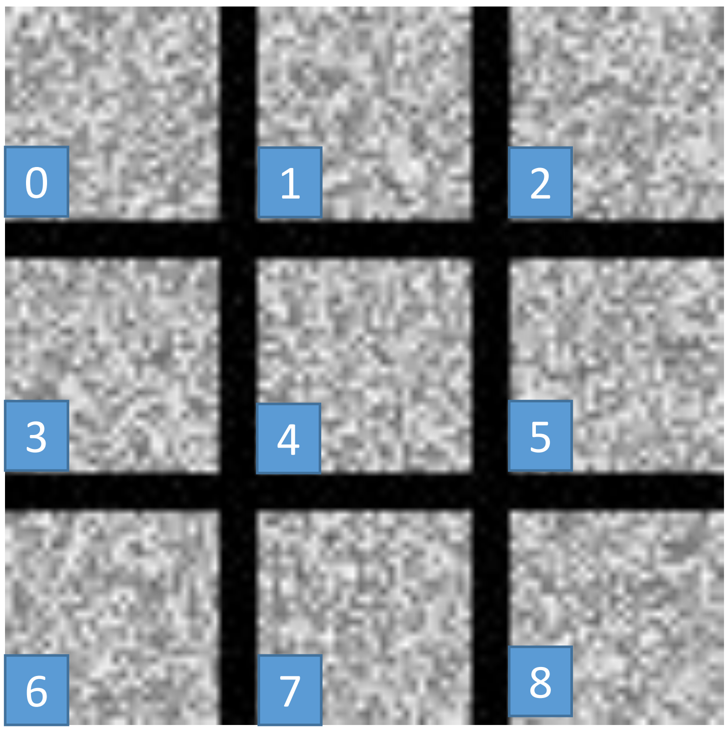
[Evolving AI Lab]

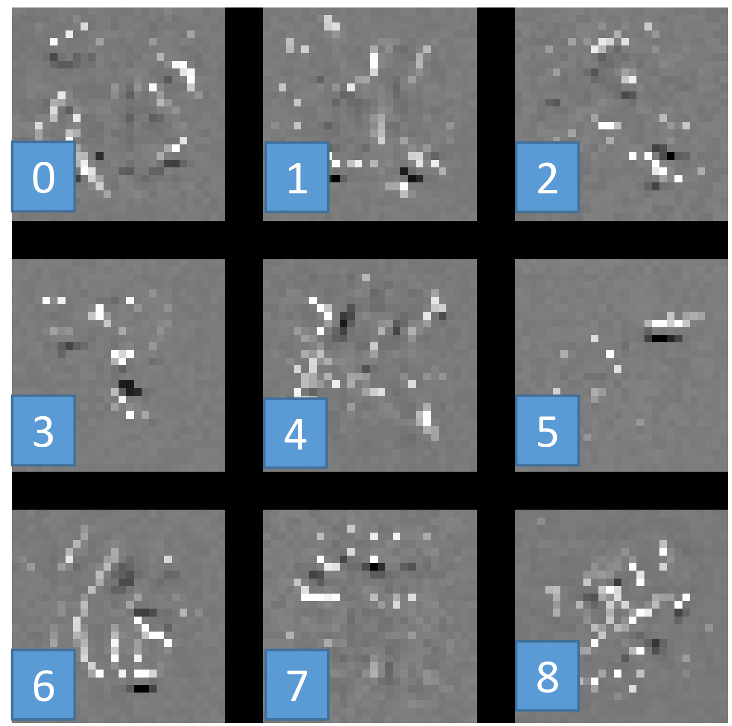
What does CNN learn?

Over all pixel values

$$x^* = arg\max_x y^i$$

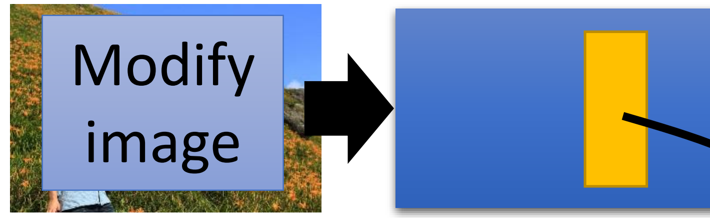
$$x^* = arg\max_x \left(y^i + \boxed{\sum_{i,j} |x_{ij}|} \right)$$





Deep Dream

CNN



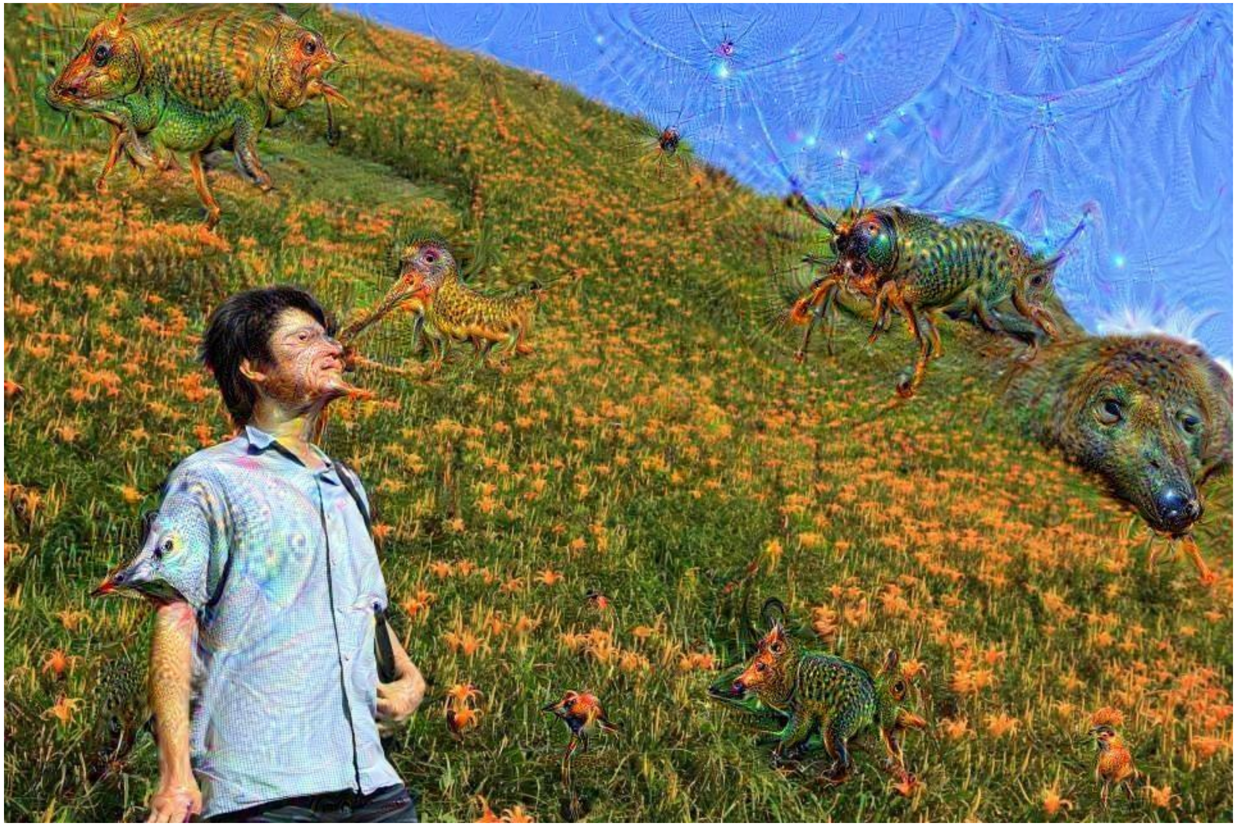
$$\begin{bmatrix} 3.9 \\ -1.5 \\ 2.3 \\ \vdots \end{bmatrix}$$

- Given a photo, machine adds what it sees ……

http://deepdreamgenerator.com/

Deep Dream

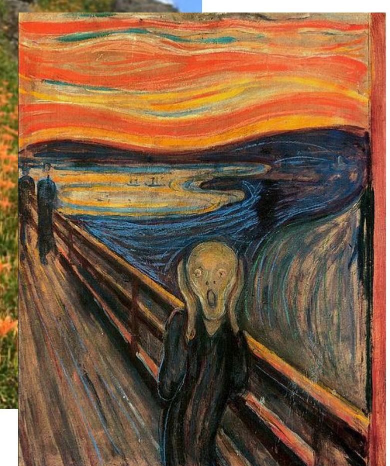
- Given a photo, machine adds what it sees ……



http://deepdreamgenerator.com/

Deep Style

- Given a photo, make its style like famous paintings



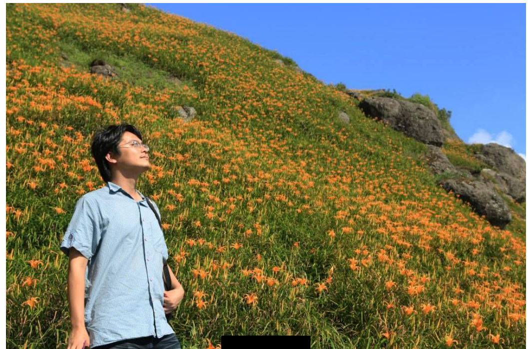
https://dreamscopeapp.com/

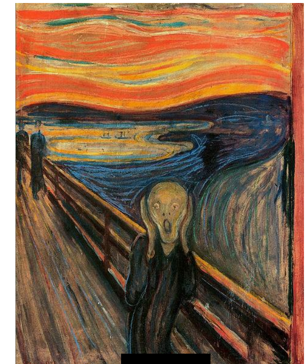
Deep Style

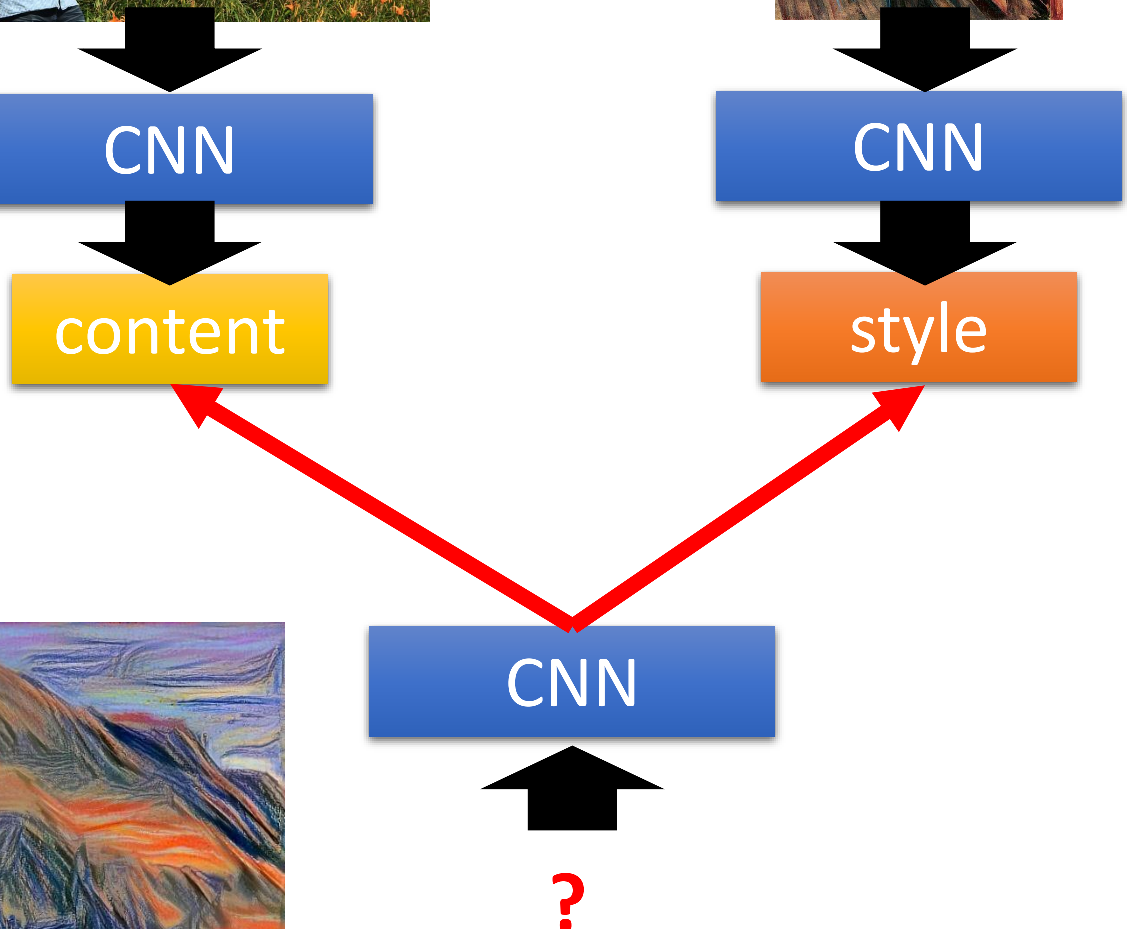
- Given a photo, make its style like famous paintings

https://dreamscopeapp.com/

Deep Style







A Neural Algorithm of Artistic Style

https://arxiv.org/abs/1508.06576

Application: Playing Go

19 x 19 matrix (image)

Network

Next move (19 x 19 positions)

19 x 19 vector

Black: 1

white: -1

none: 0

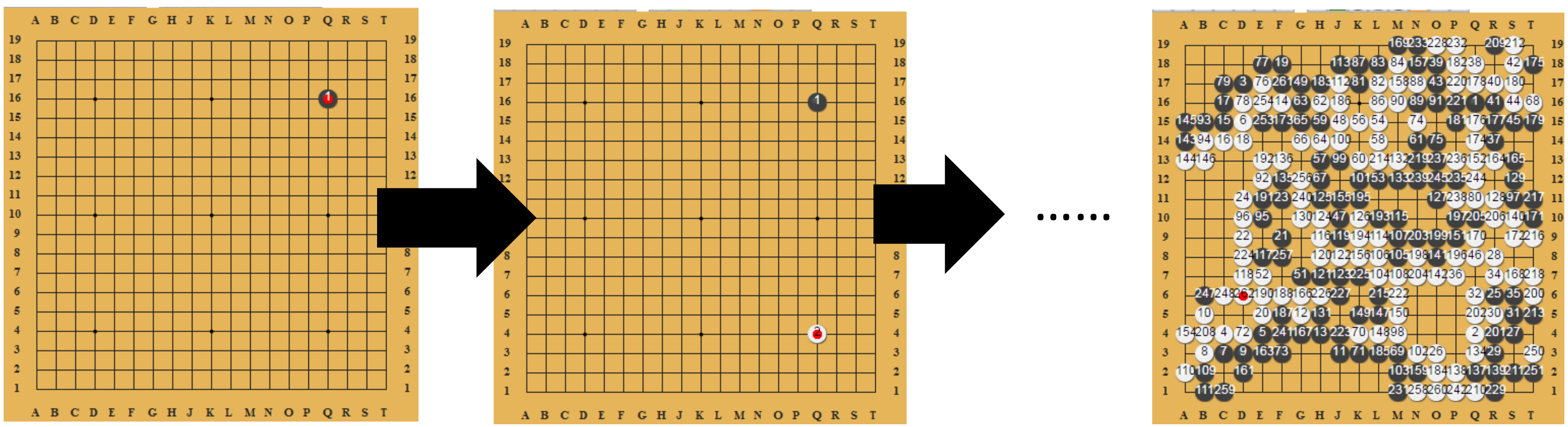
Fully-connected feedforward network can be used

But CNN performs much better.

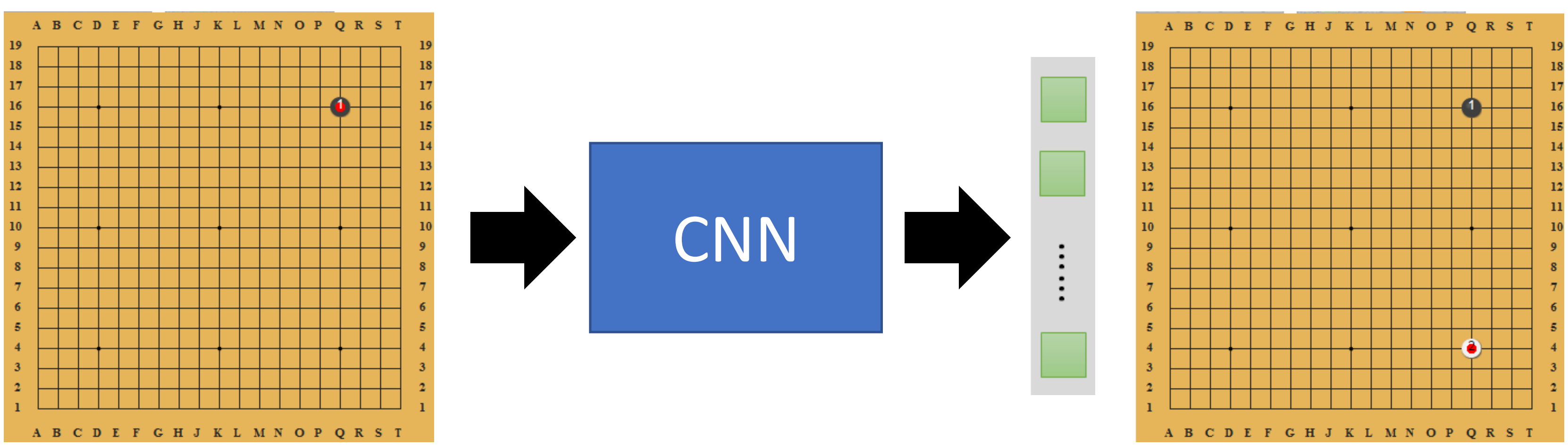
Training

http://lgs.tw/qwwheue

Collecting records of many previous plays



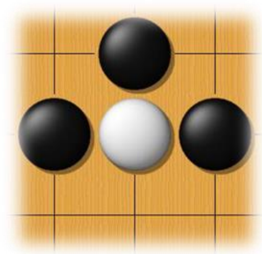
Machine mimics human player



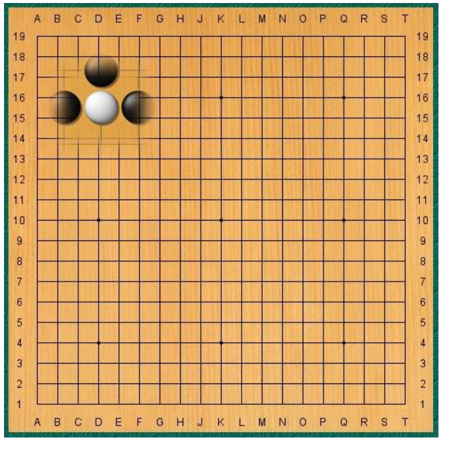
Why CNN for Go playing?

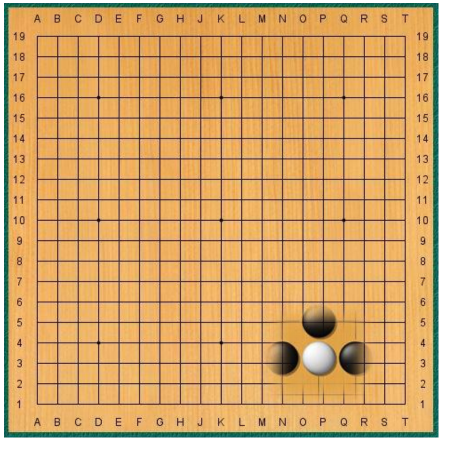
- Some patterns are much smaller than the whole image

Alpha Go uses 5 x 5 for first layer



- The same patterns appear in different regions.





Why CNN for Go playing?

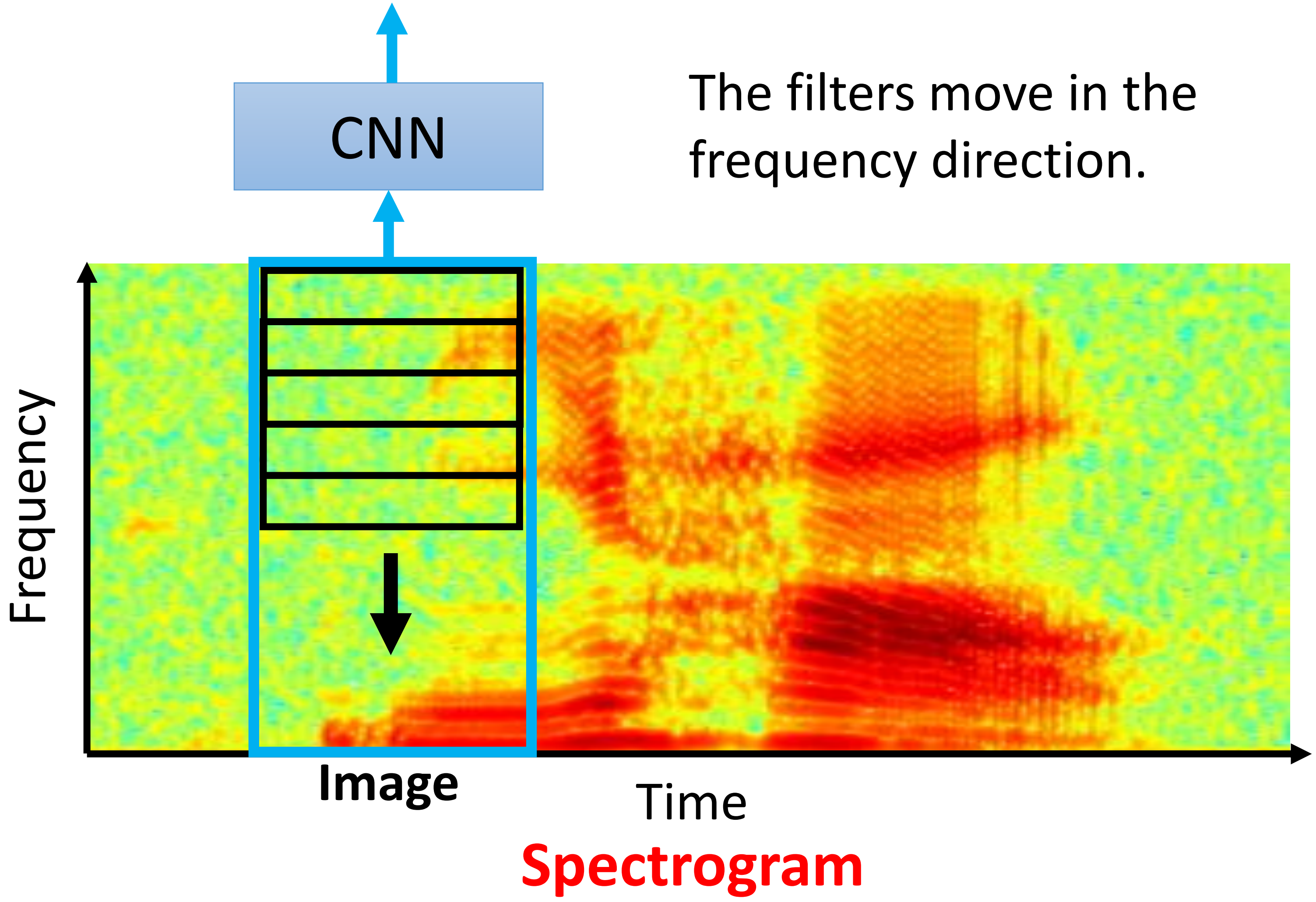
- Subsampling the pixels will not change the object

→ Max Pooling How to explain this???

Neural network architecture. The input to the policy network is a 19 × 19 × 48 image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23 × 23 image, then convolves *k* filters of kernel size 5 × 5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21 × 21 image, then convolves *k* filters of kernel size 3 × 3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1 × 1 with strid[illegible]max func- tion. The [illegible] Extended Data Table 3 additionally show the results of training with *k* = 128, 256 and 384 filters.

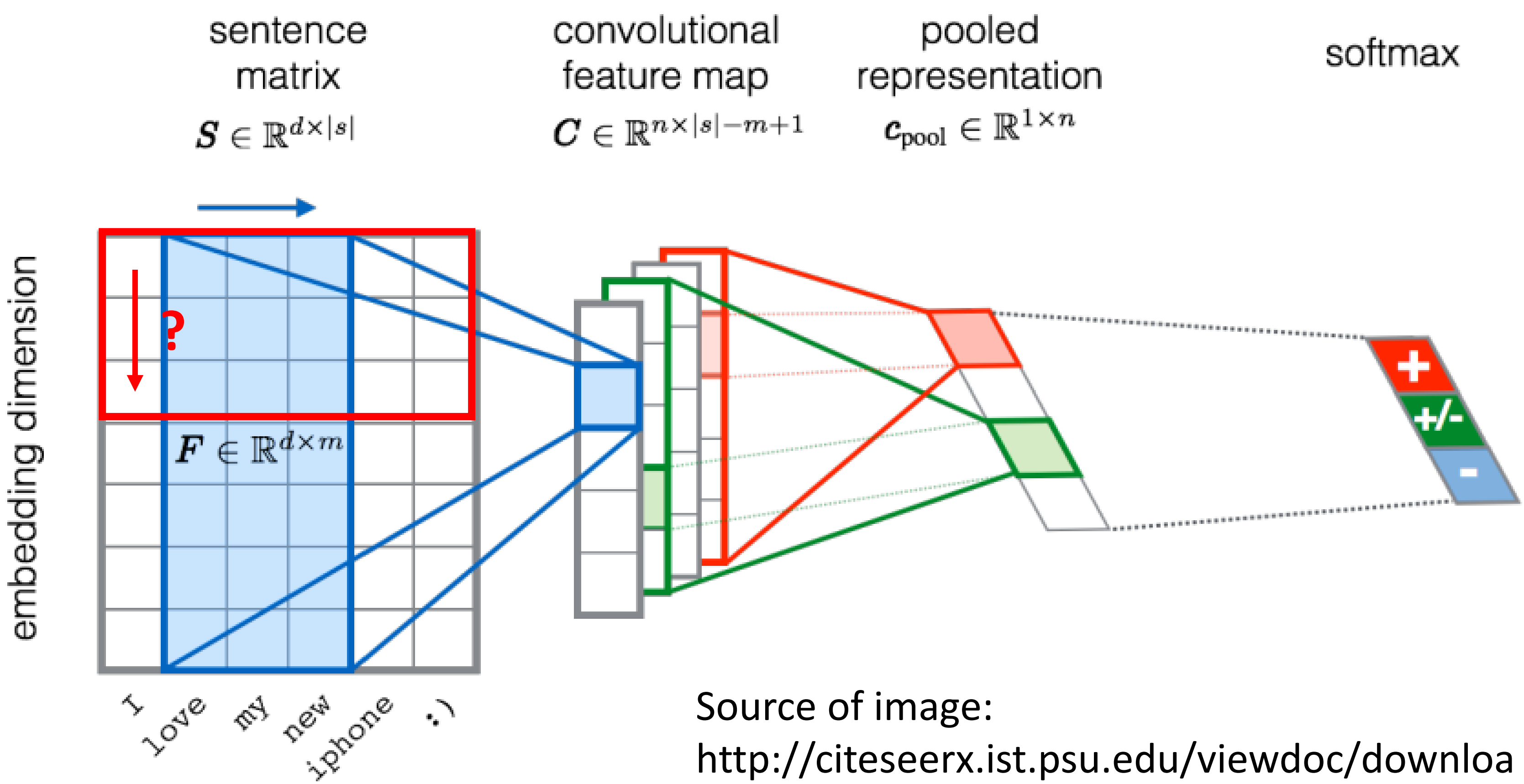
Alpha Go does not use Max Pooling ……

More Application: Speech



The filters move in the frequency direction.

More Application: Text



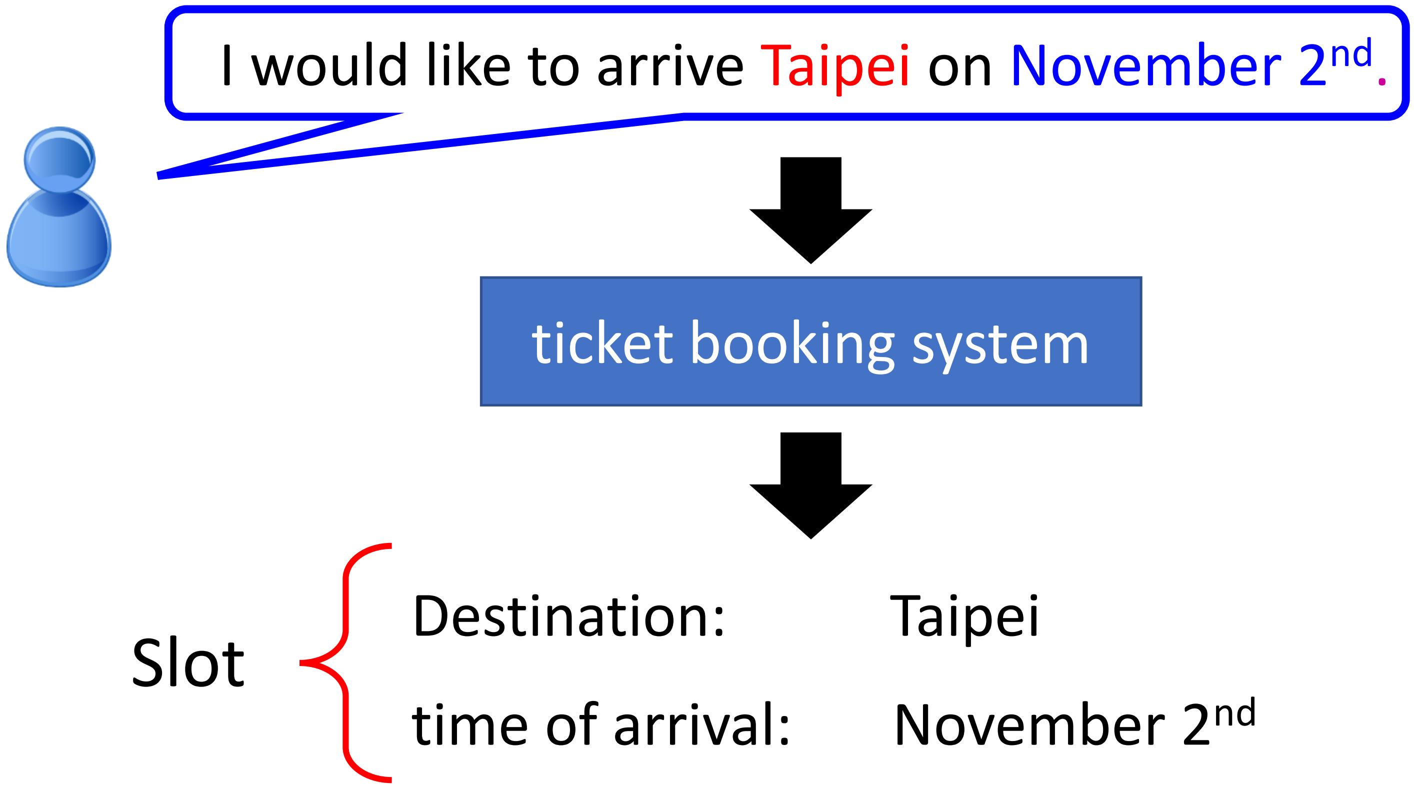
Source of image:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.703.6858&rep=rep1&type=pdf

Lecture V:
Recurrent Neural Network (RNN)

Neural Network with Memory

Example Application

- Slot Filling

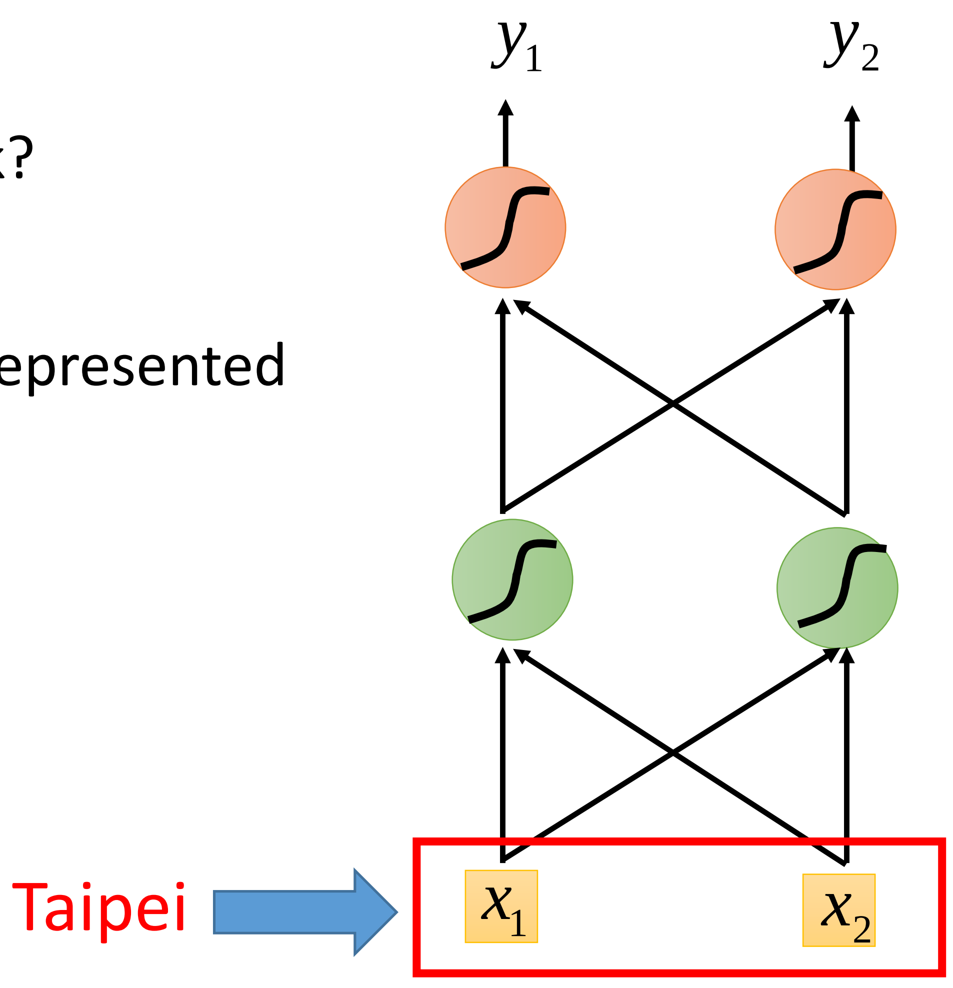


Example Application

Solving slot filling by Feedforward network?

Input: a word

(Each word is represented as a vector)



1-of-N encoding

How to represent each word as a vector?

1-of-N Encoding lexicon = {apple, bag, cat, dog, elephant}

The vector is lexicon size.

Each dimension corresponds to a word in the lexicon

The dimension for the word is 1, and others are 0

apple = [1 0 0 0 0]

bag = [0 1 0 0 0]

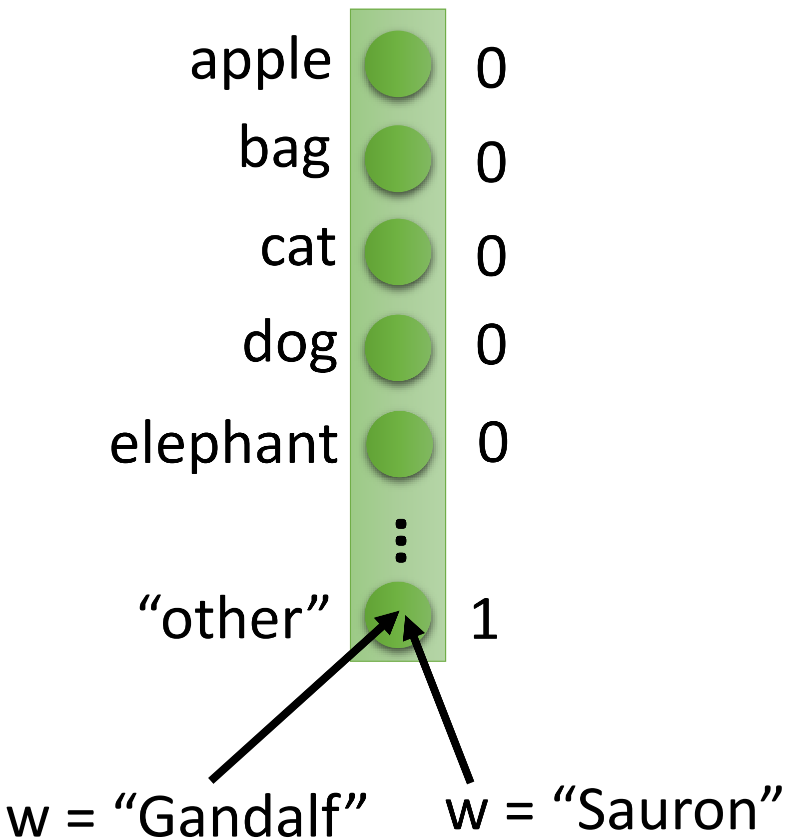
cat = [0 0 1 0 0]

dog = [0 0 0 1 0]

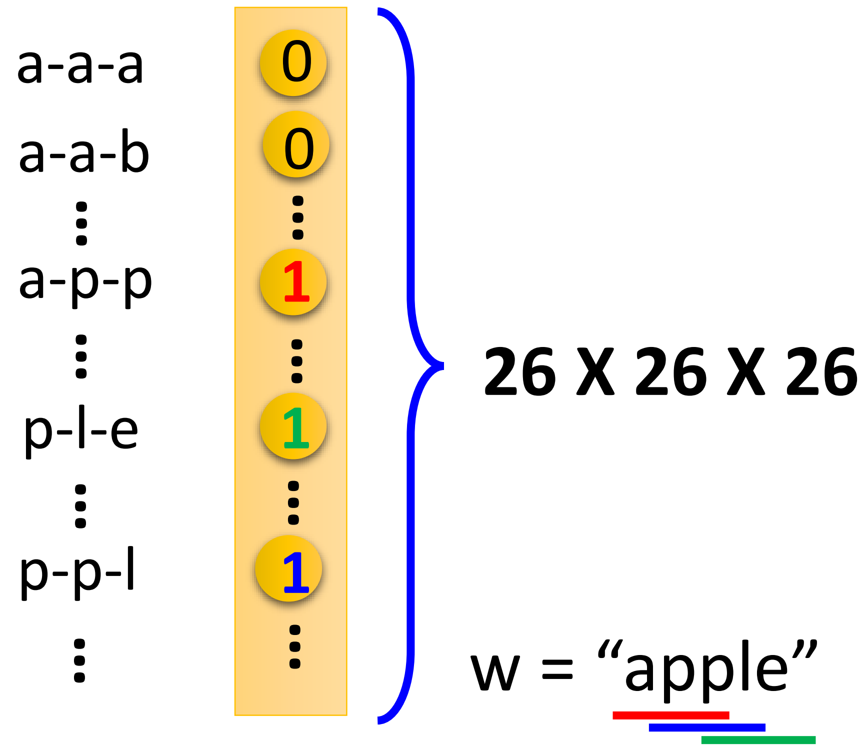
elephant = [0 0 0 0 1]

Beyond 1-of-N encoding

Dimension for "Other"



Word hashing



Example Application

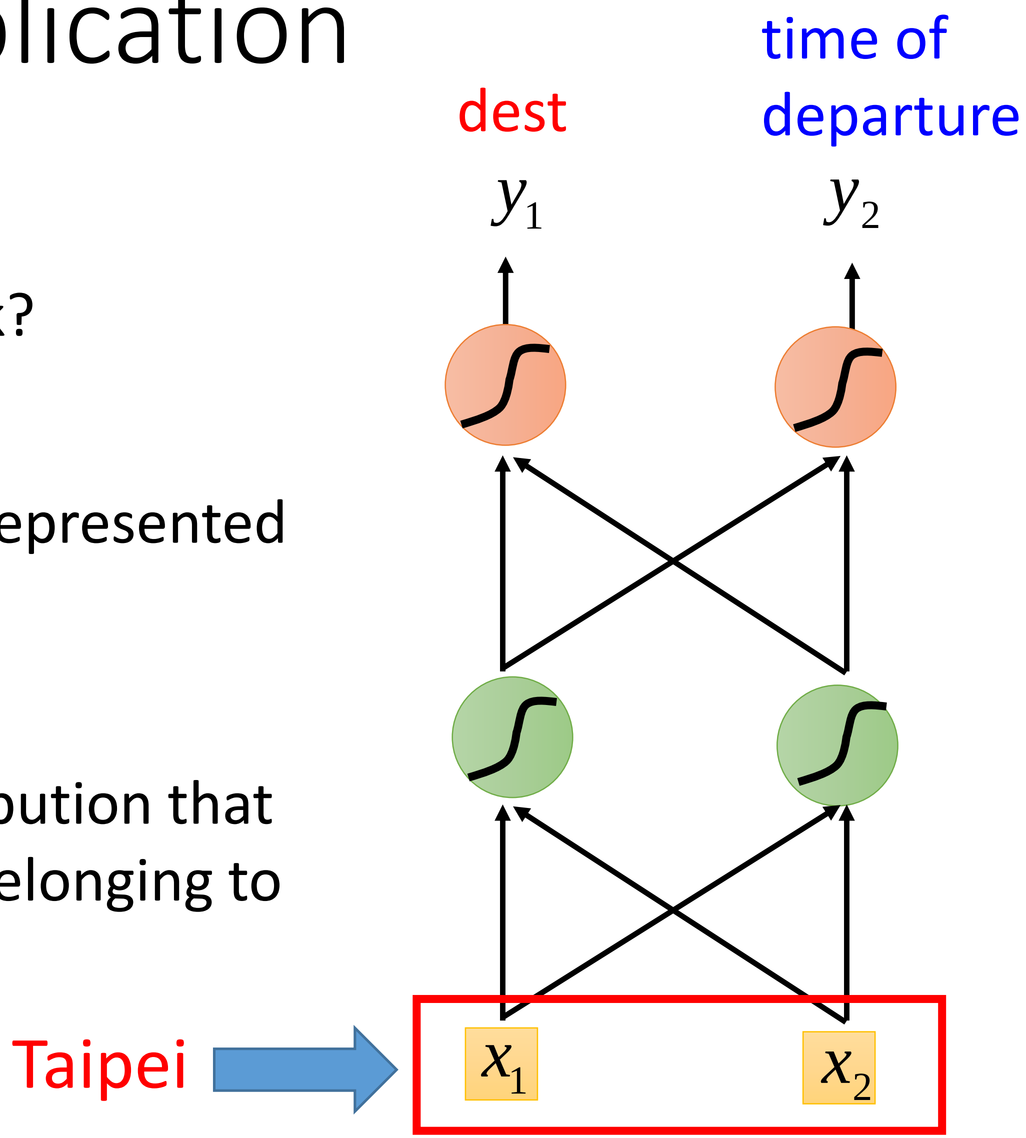
Solving slot filling by Feedforward network?

Input: a word

(Each word is represented as a vector)

Output:

Probability distribution that the input word belonging to the slots



Example Application

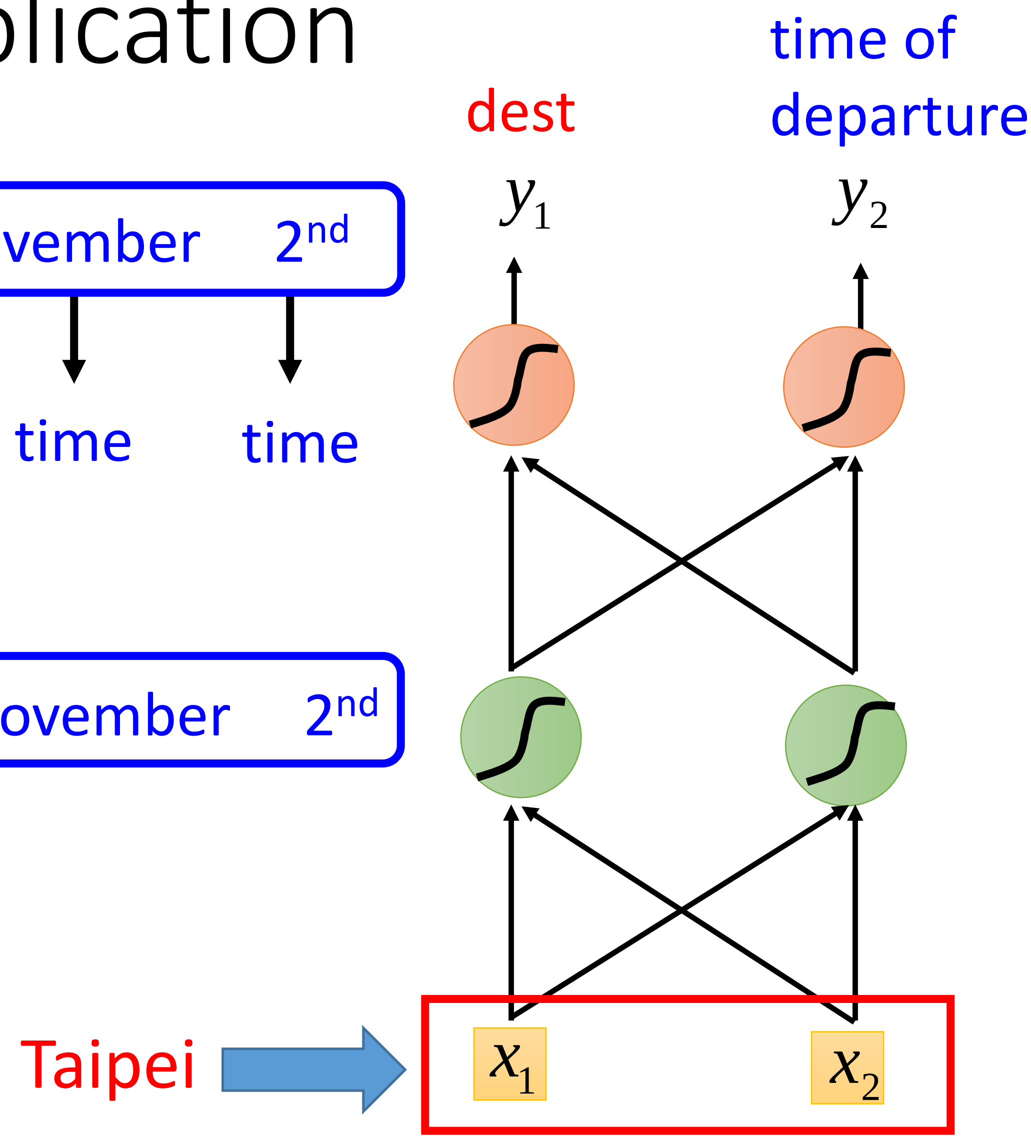
arrive	Taipei	on	November	2nd
other	dest	other	time	time

Problem?

leave Taipei on November 2nd

Taipei → place of departure

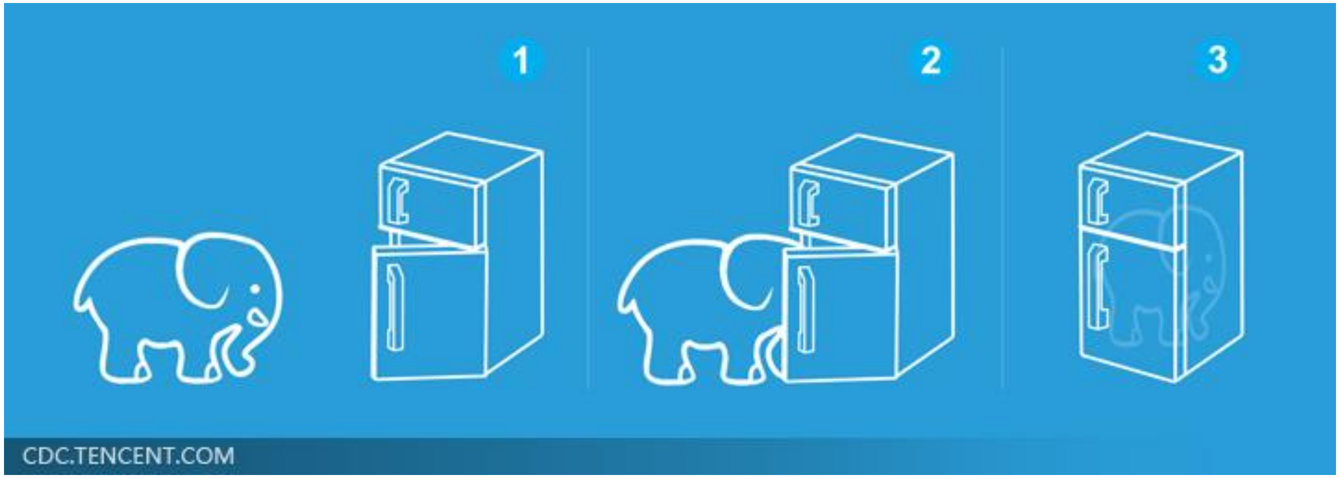
Neural network needs memory!



Three Steps for Deep Learning

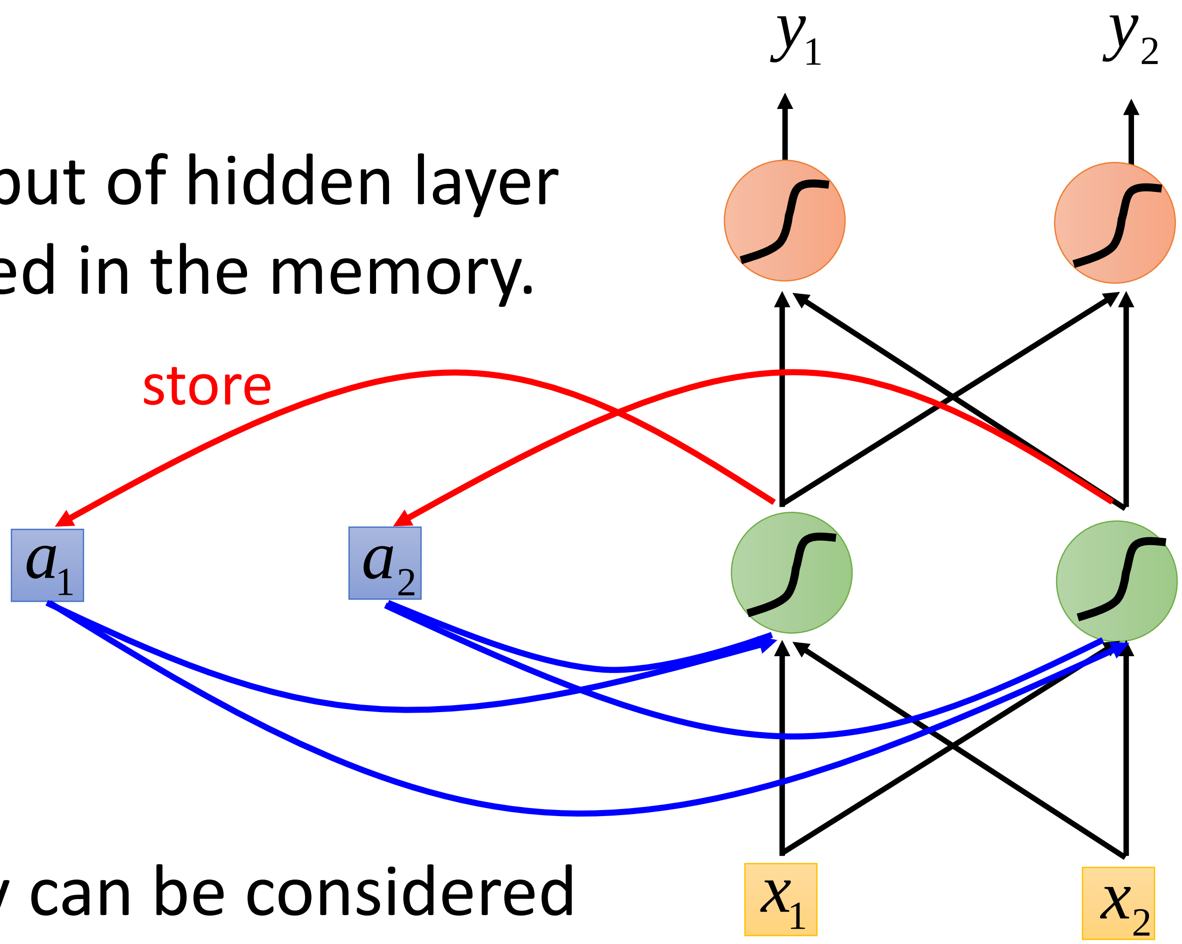
Step 1: Recurrent Neural Network → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……



Recurrent Neural Network (RNN)

The output of hidden layer are stored in the memory.

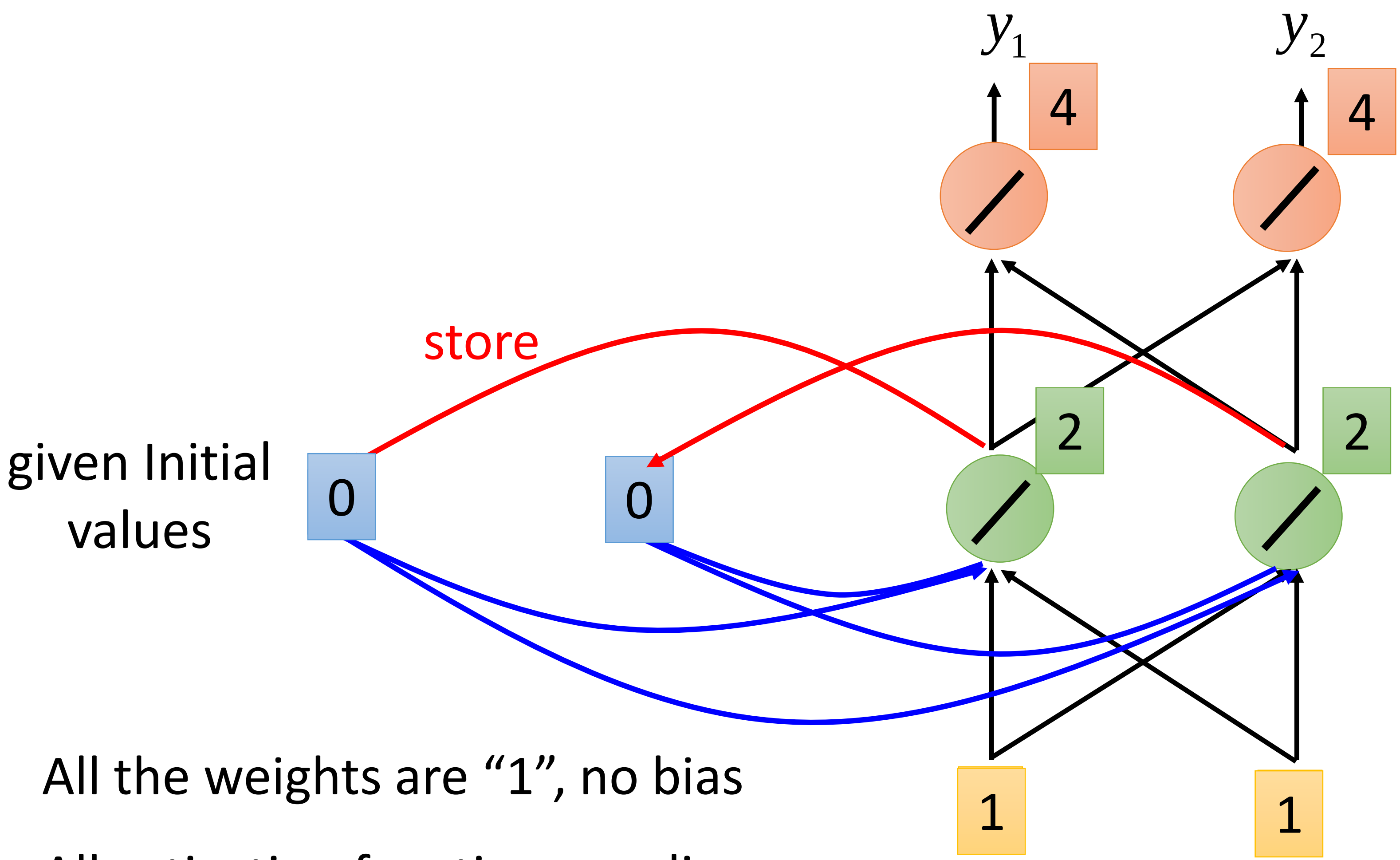


Memory can be considered as another input.

Example

Input sequence: $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}2\\2\end{bmatrix}$ ……

output sequence: $\begin{bmatrix}4\\4\end{bmatrix}$



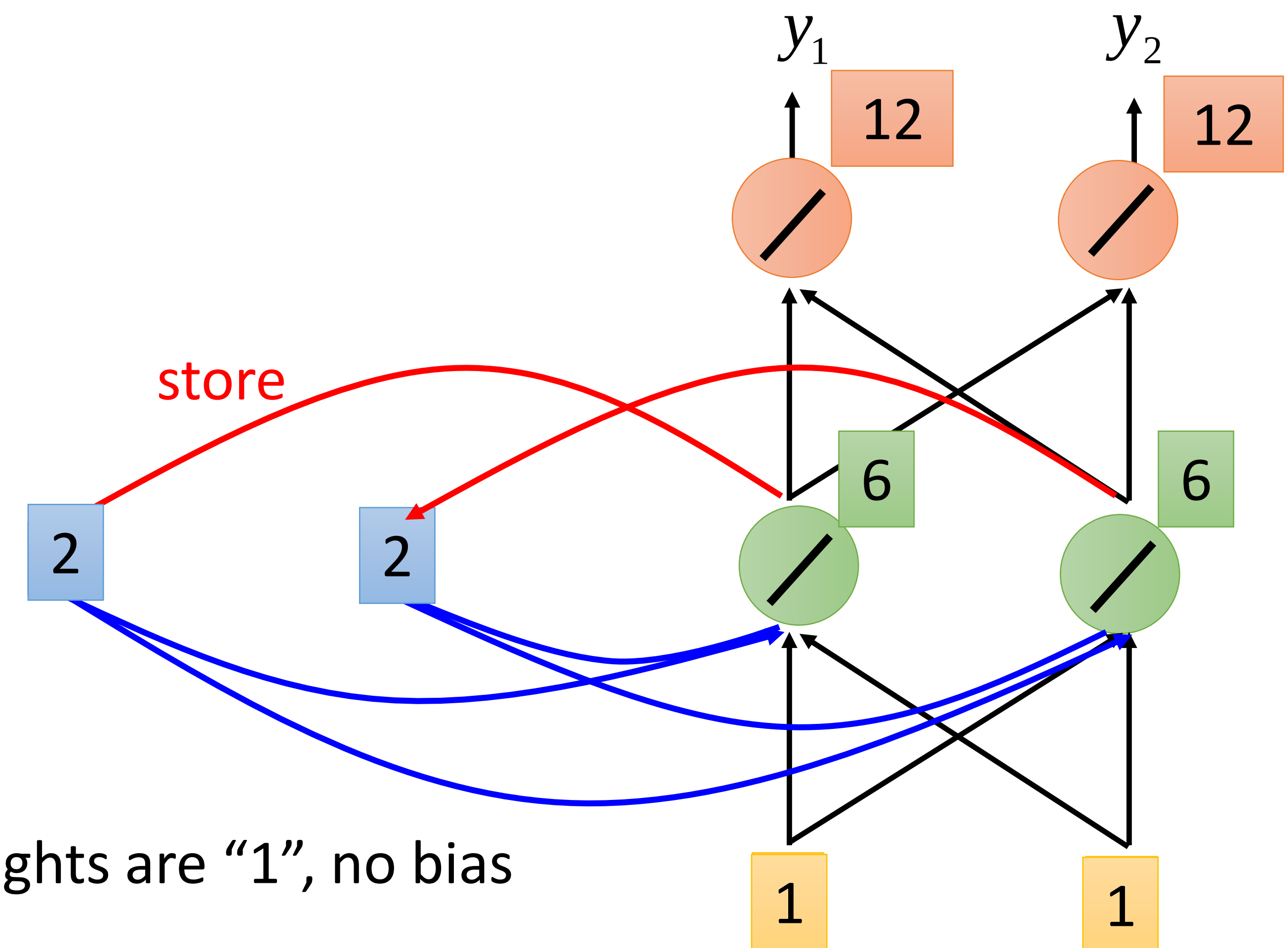
All the weights are "1", no bias

All activation functions are linear

Example

Input sequence: $\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}2\\2\end{bmatrix}$ ……

output sequence: $\begin{bmatrix}4\\4\end{bmatrix}\begin{bmatrix}12\\12\end{bmatrix}$



All the weights are "1", no bias

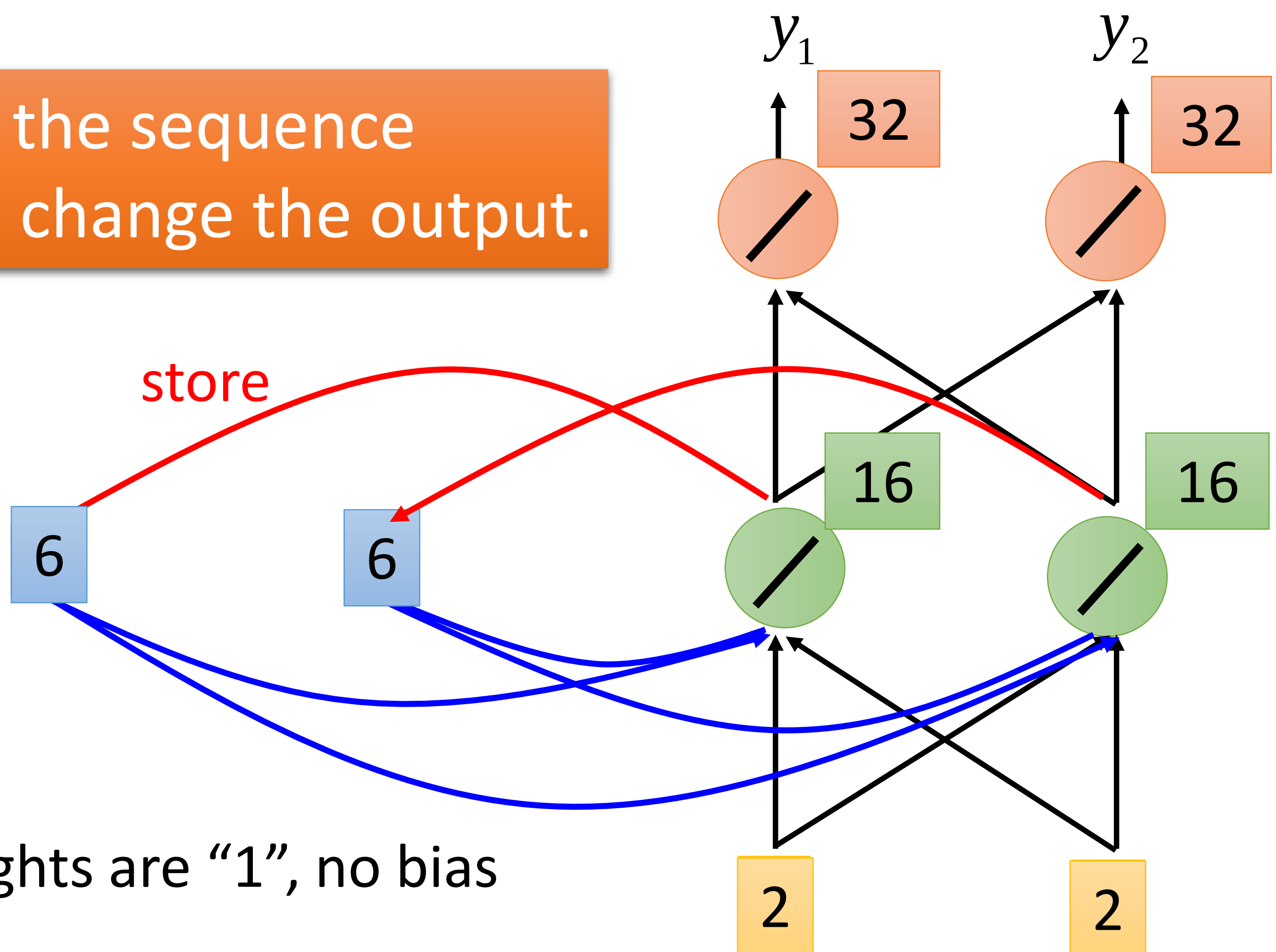
All activation functions are linear

Example

Input sequence: $\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}2\\2\end{bmatrix}$ ……

output sequence: $\begin{bmatrix}4\\4\end{bmatrix}\begin{bmatrix}12\\12\end{bmatrix}\begin{bmatrix}32\\32\end{bmatrix}$

Changing the sequence order will change the output.

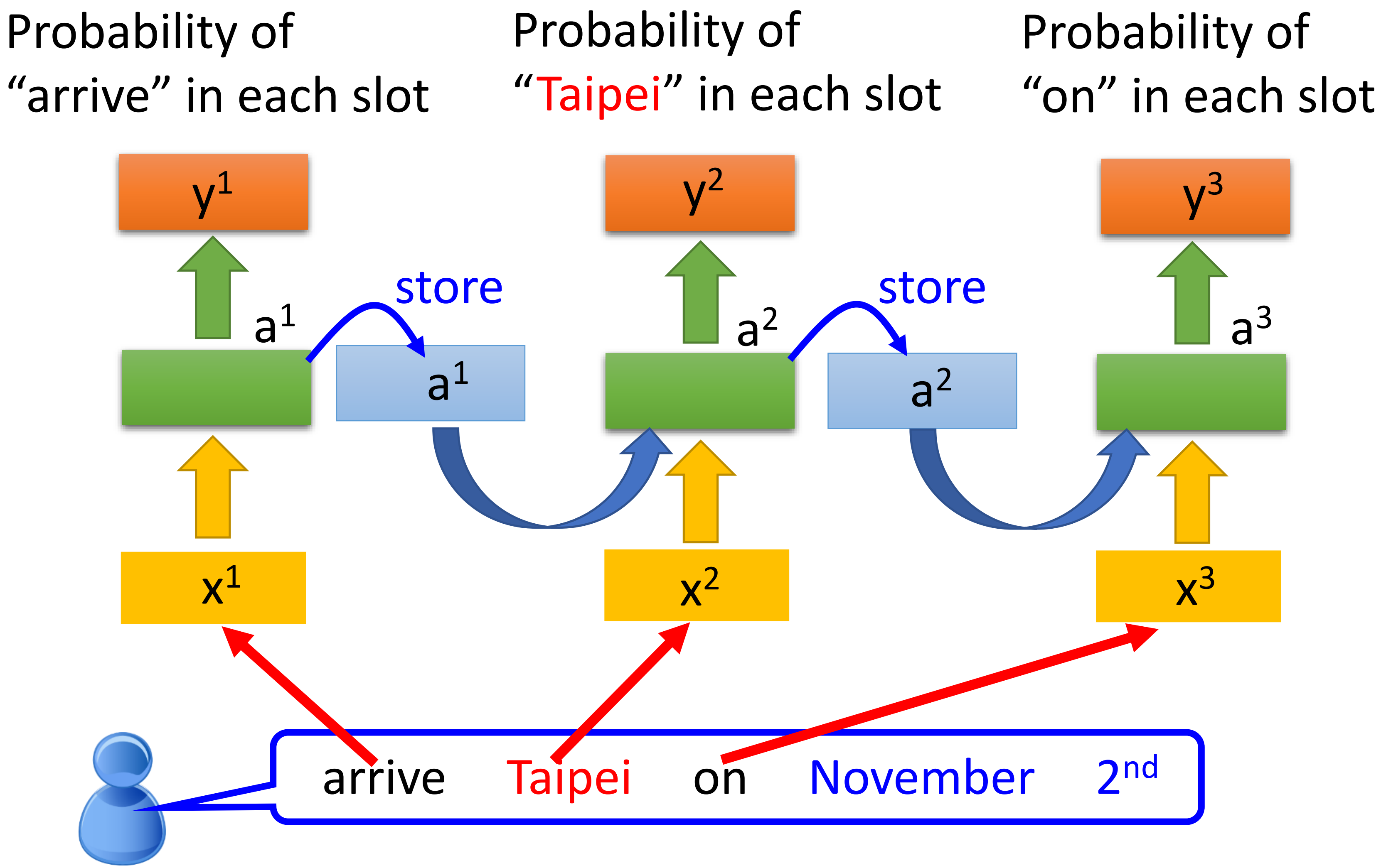


All the weights are "1", no bias

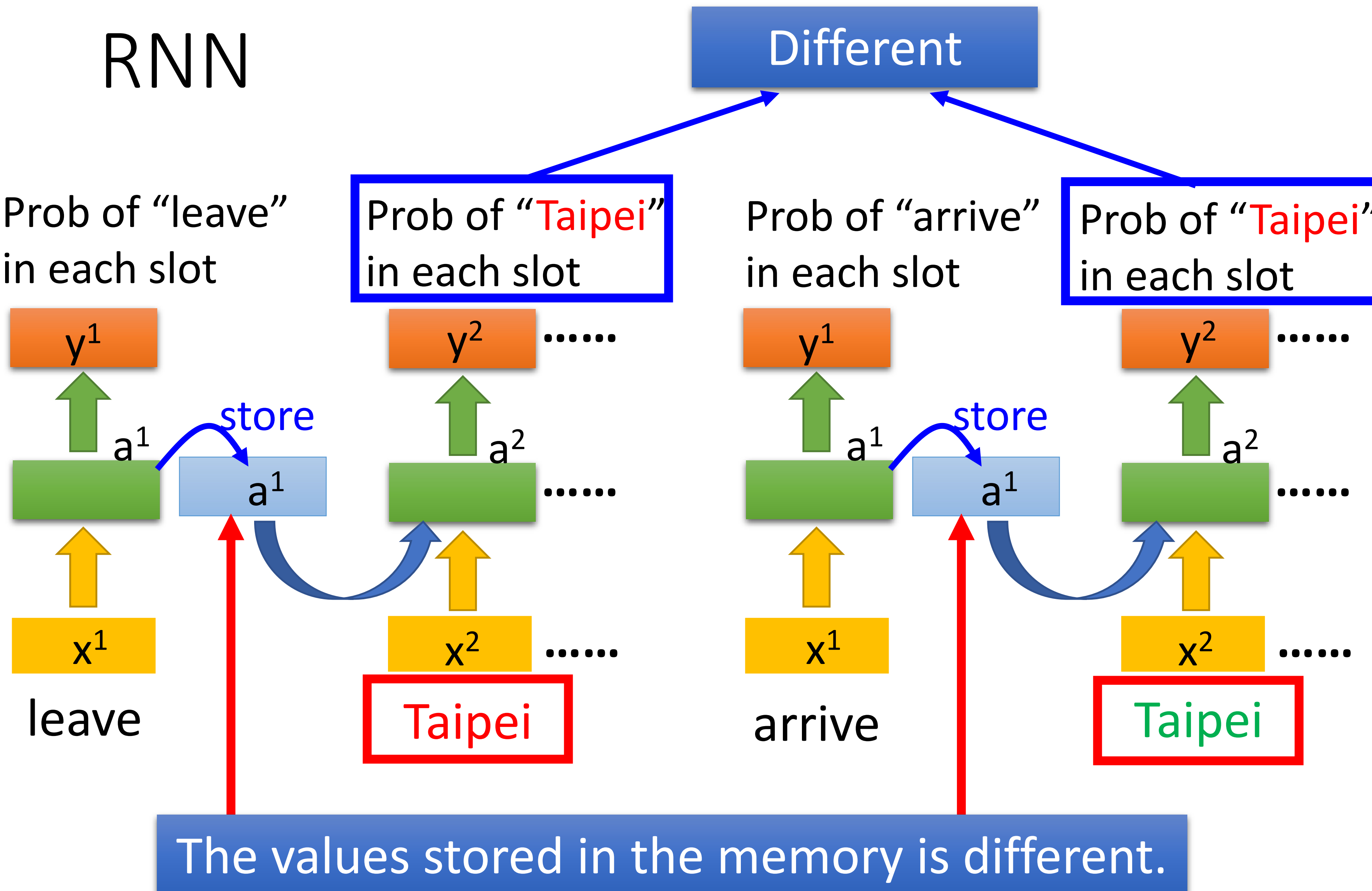
All activation functions are linear

RNN

The same network is used again and again.

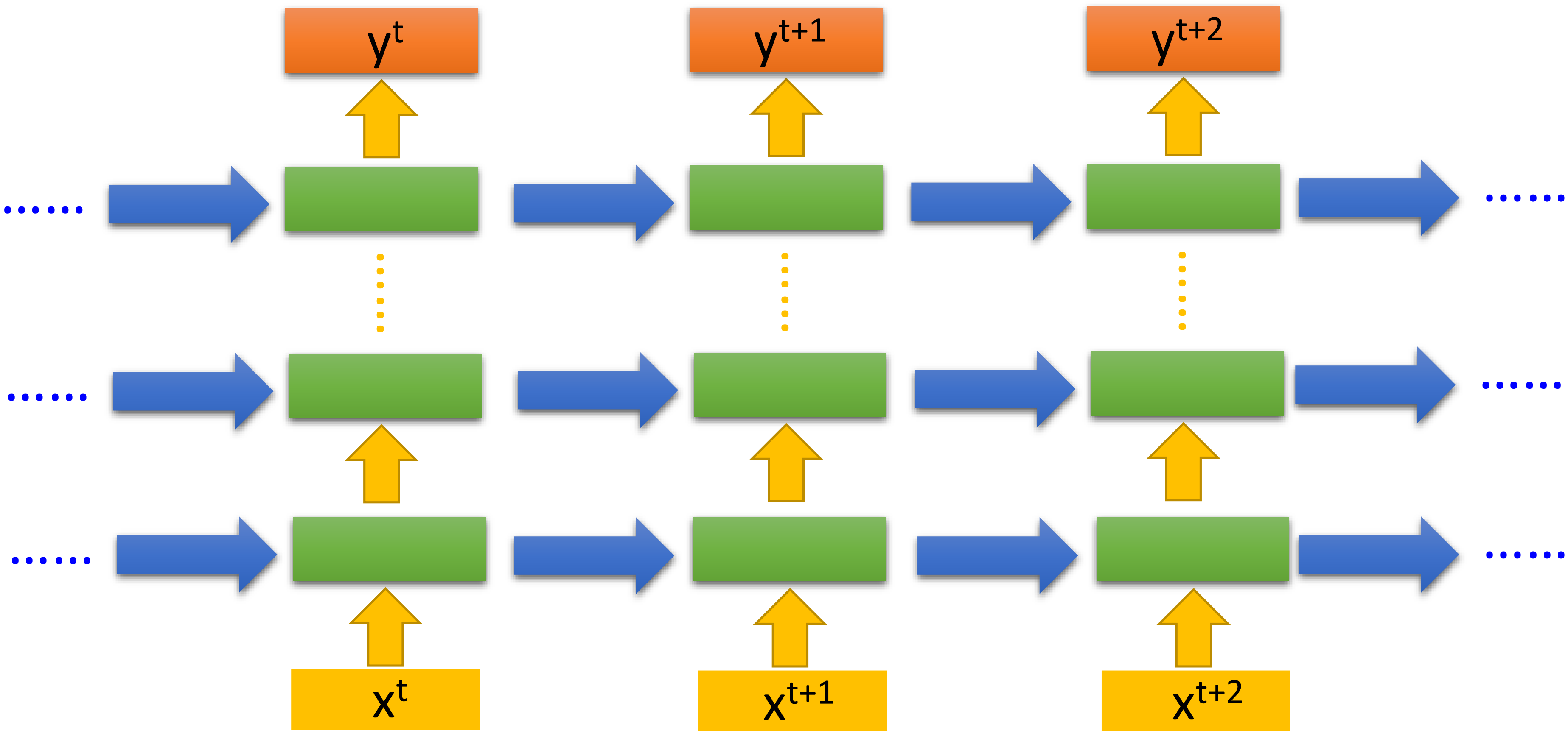


RNN

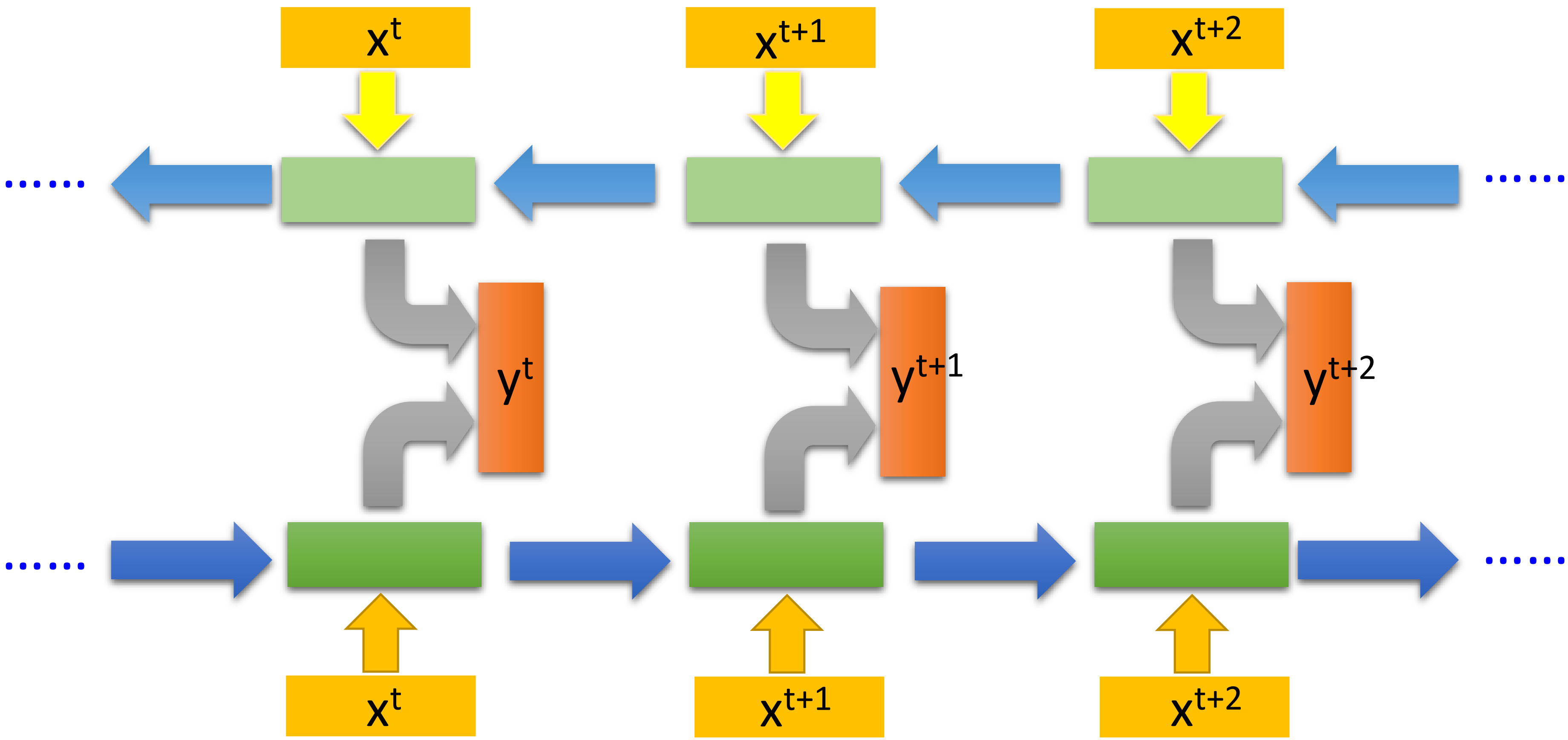


Different
Prob of "leave" in each slot
Prob of "Taipei" in each slot
Prob of "arrive" in each slot
Prob of "Taipei" in each slot
y1
y2
......
y1
y2
......
a1
store
a1
a2
......
a1
store
a1
a2
......
x1
x2
......
x1
x2
......
leave
Taipei
arrive
Taipei
The values stored in the memory is different.

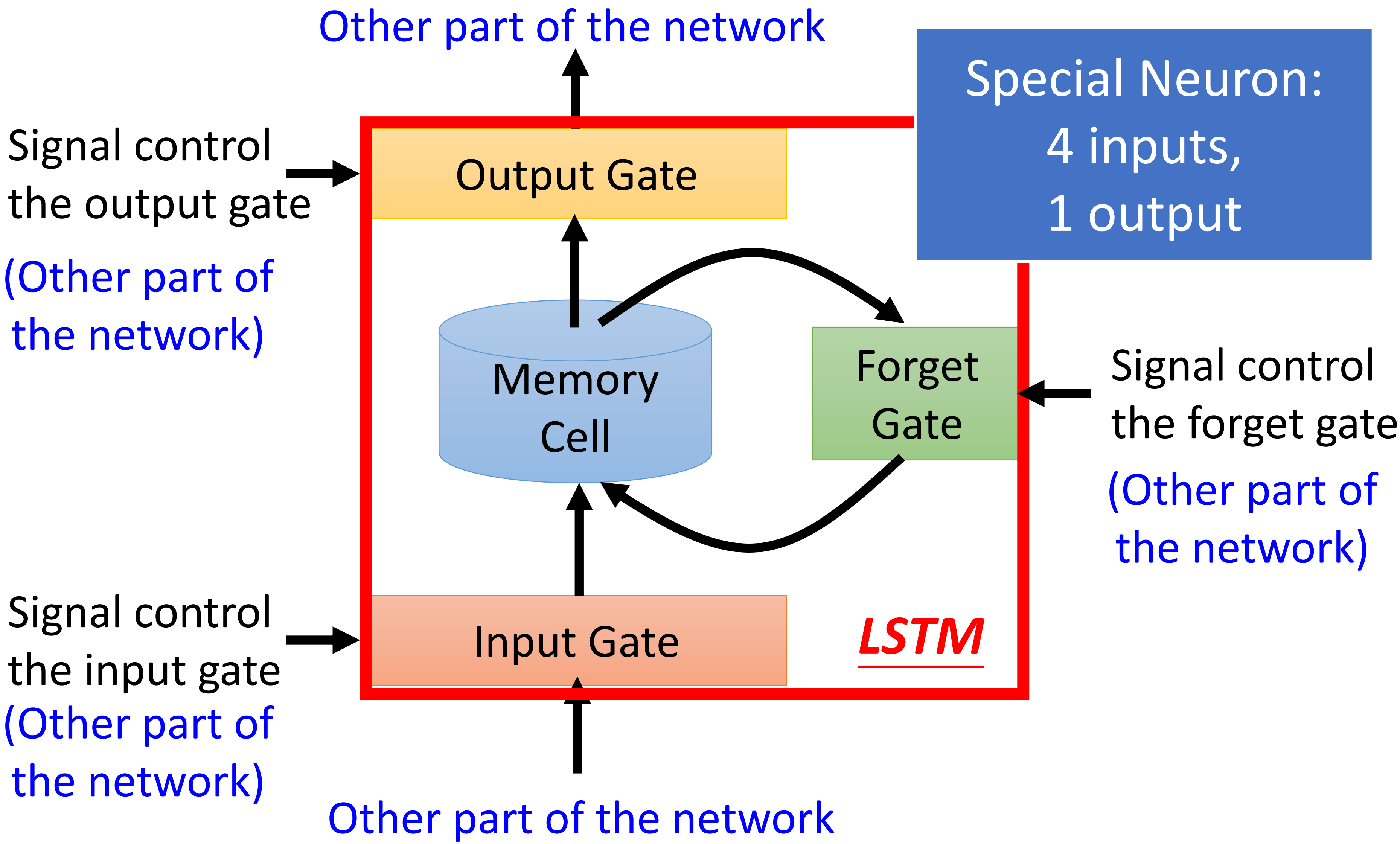
Of course it can be deep …

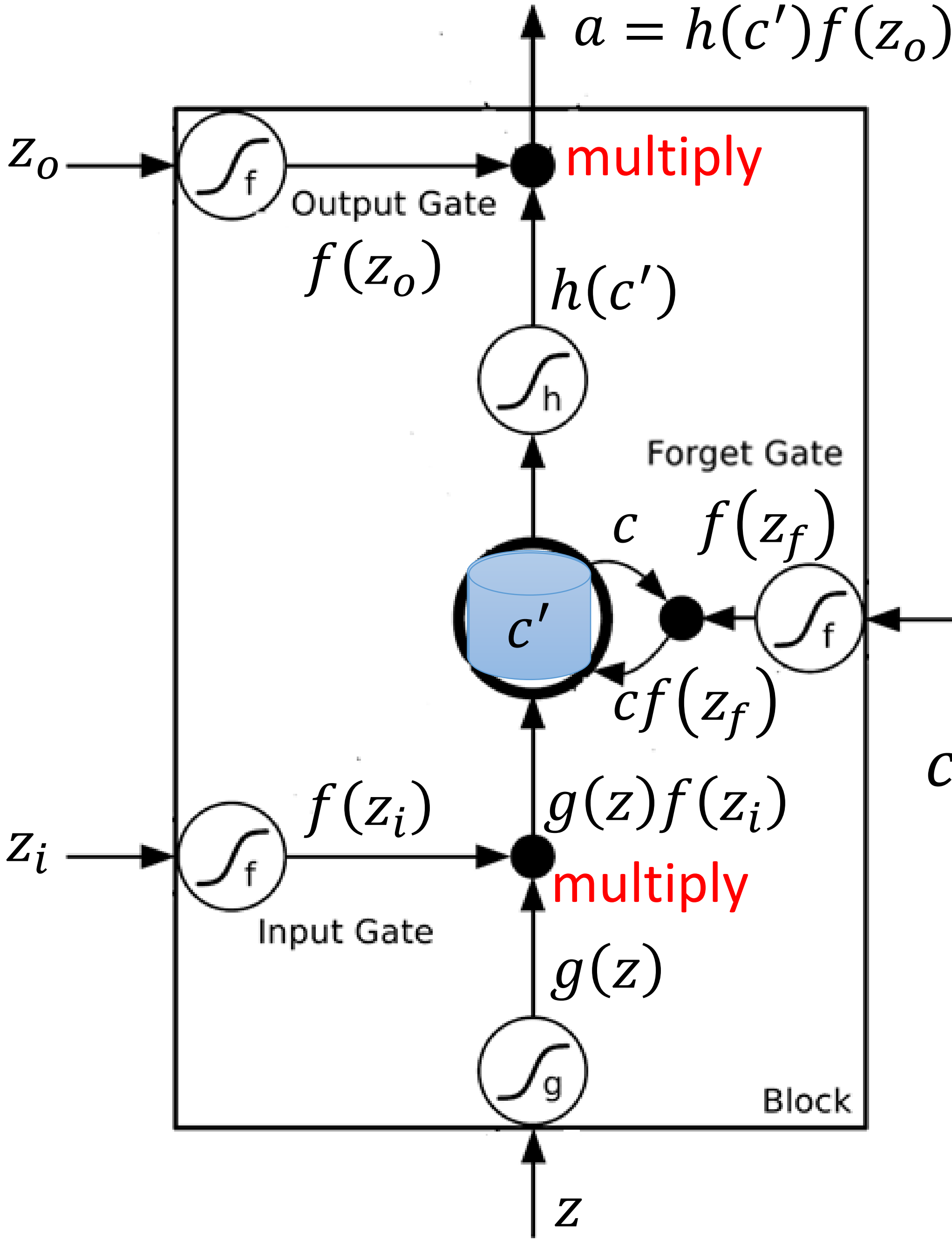


Bidirectional RNN



Long Short-term Memory (LSTM)



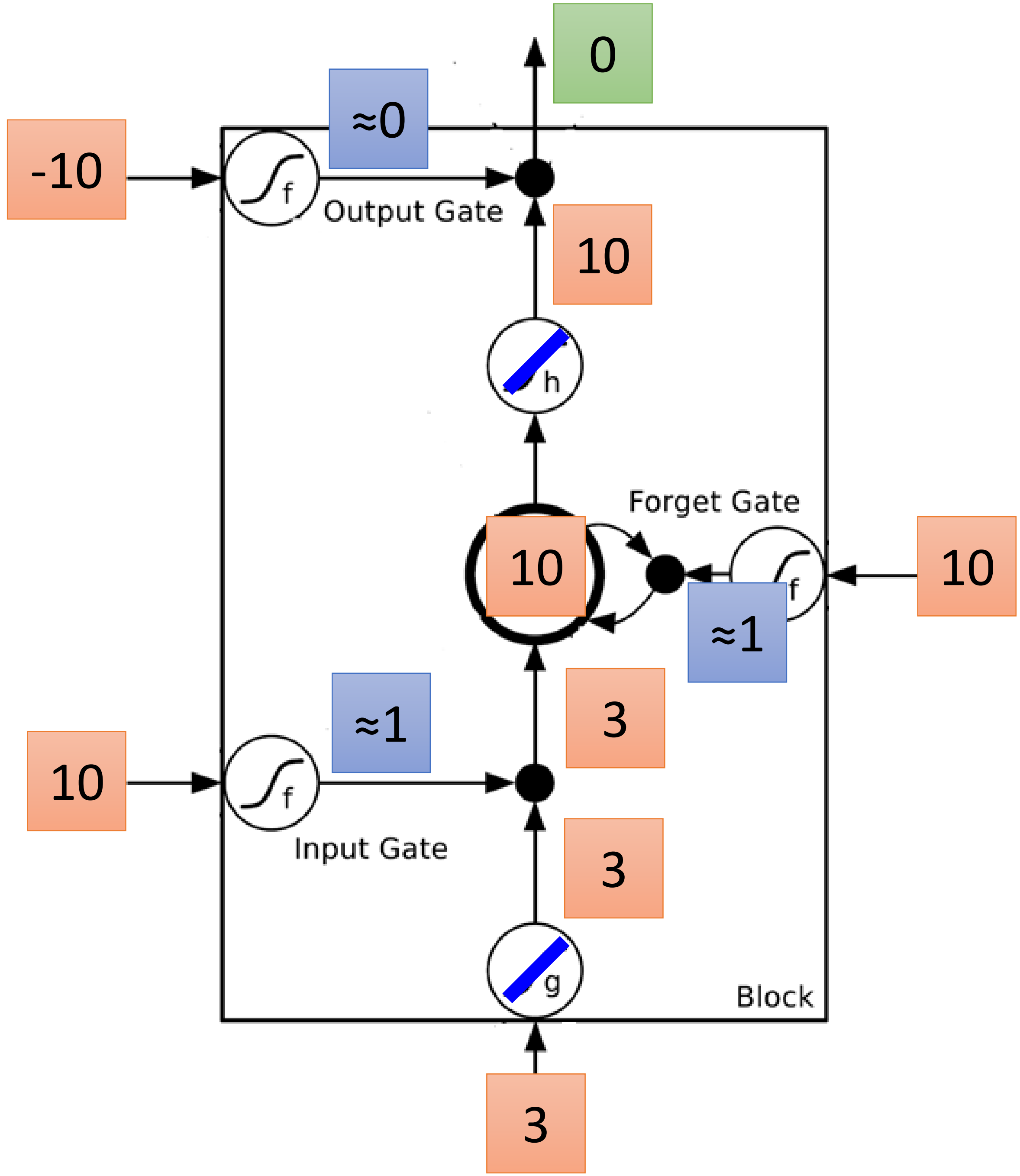


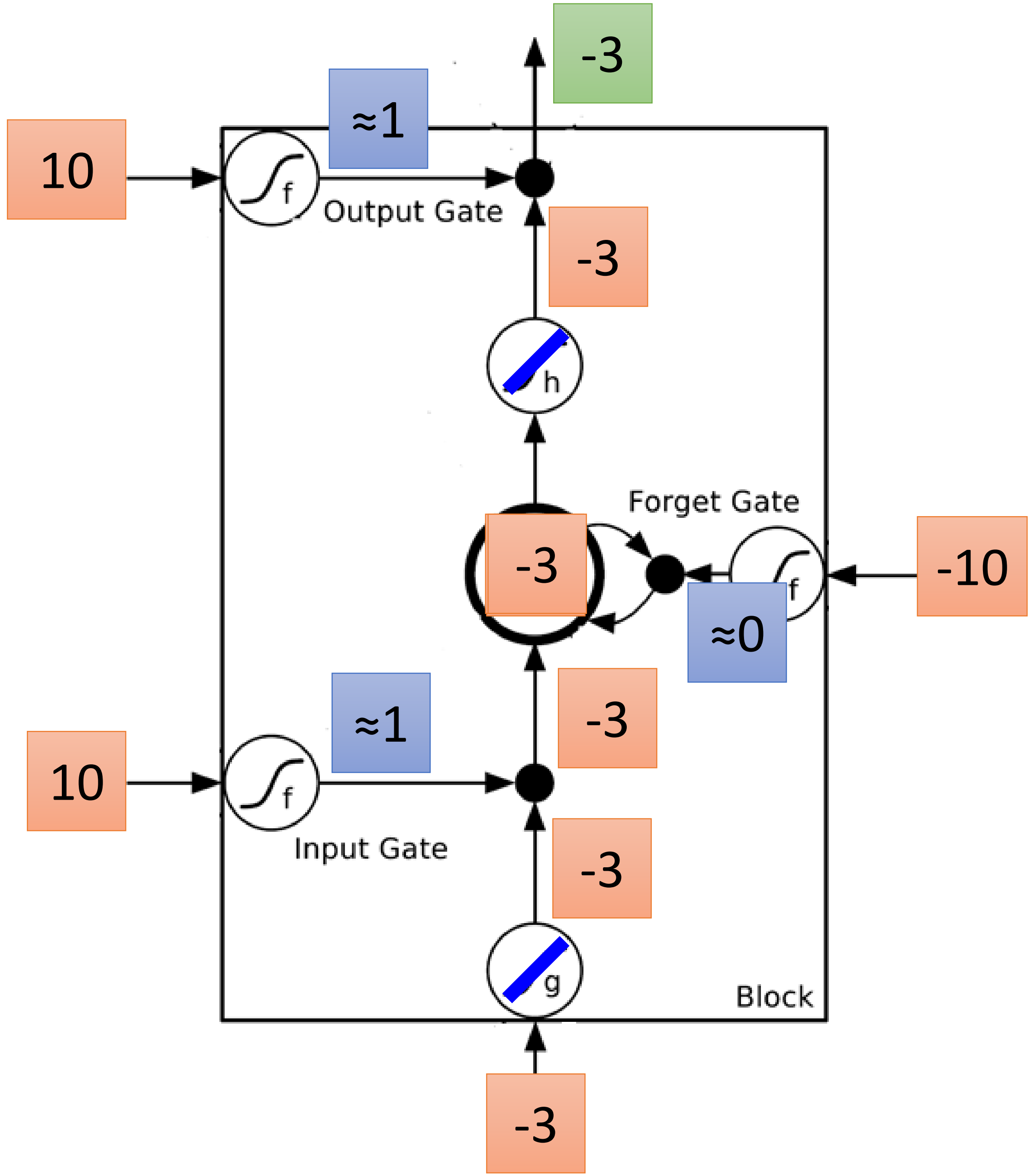
Activation function f is usually a sigmoid function

Between 0 and 1

Mimic open and close gate

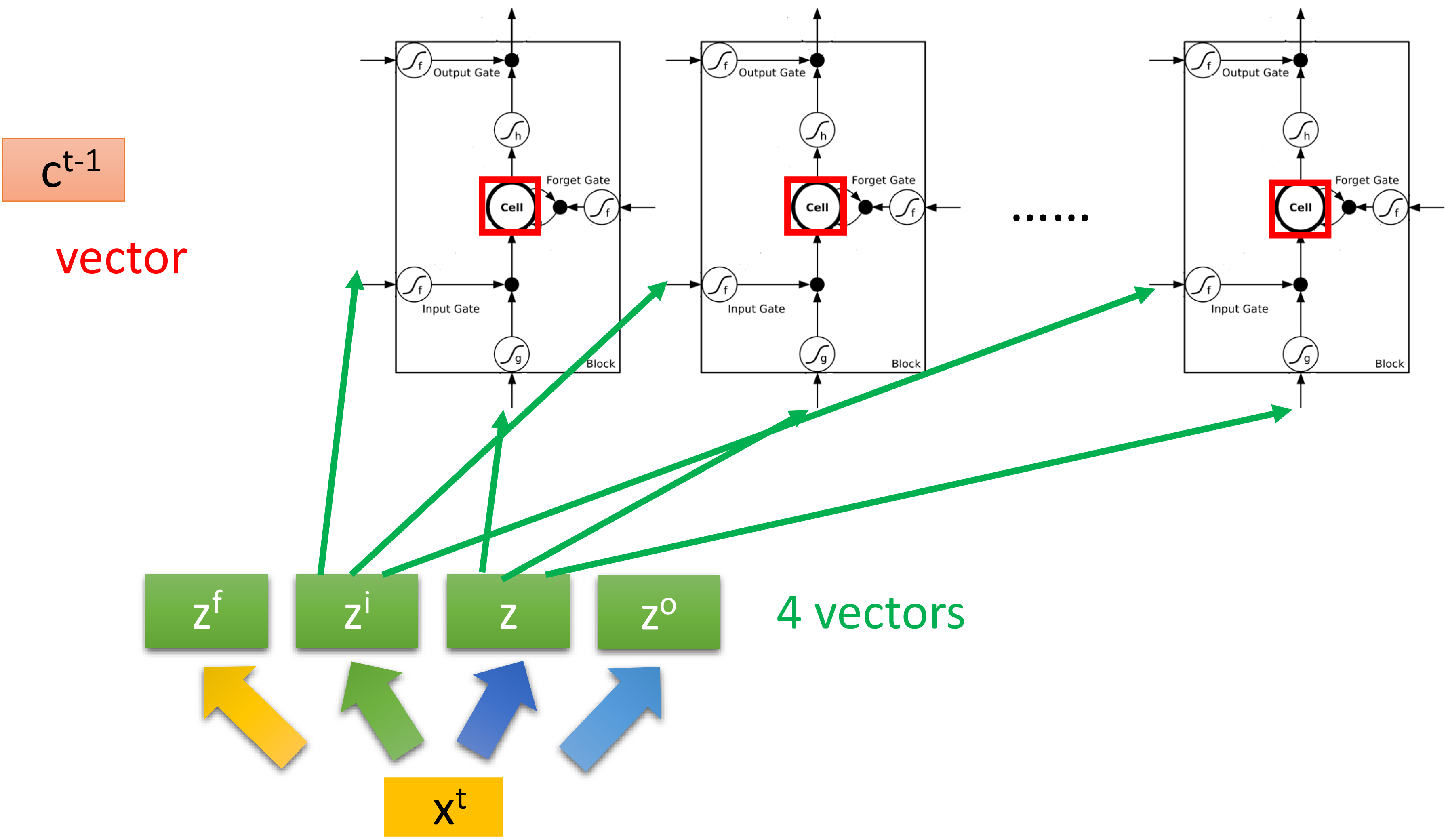
$$c' = g(z)f(z_i) + cf(z_f)$$


0
≈0
-10
Output Gate
10
h
Forget Gate
10
10
≈1
≈1
3
10
Input Gate
3
g
Block
3



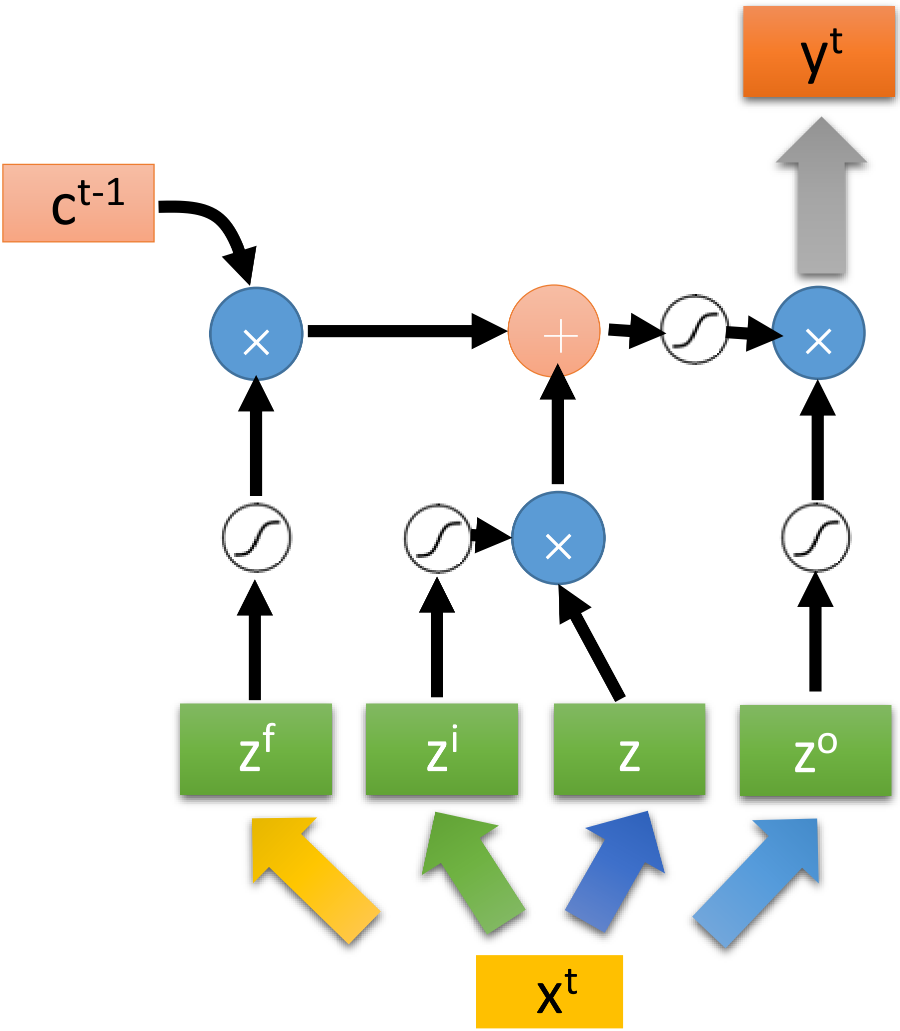
-3
10
≈1
Output Gate
-3
h
Forget Gate
-3
-10
≈0
f
≈1
-3
10
Input Gate
-3
g
Block
-3

LSTM

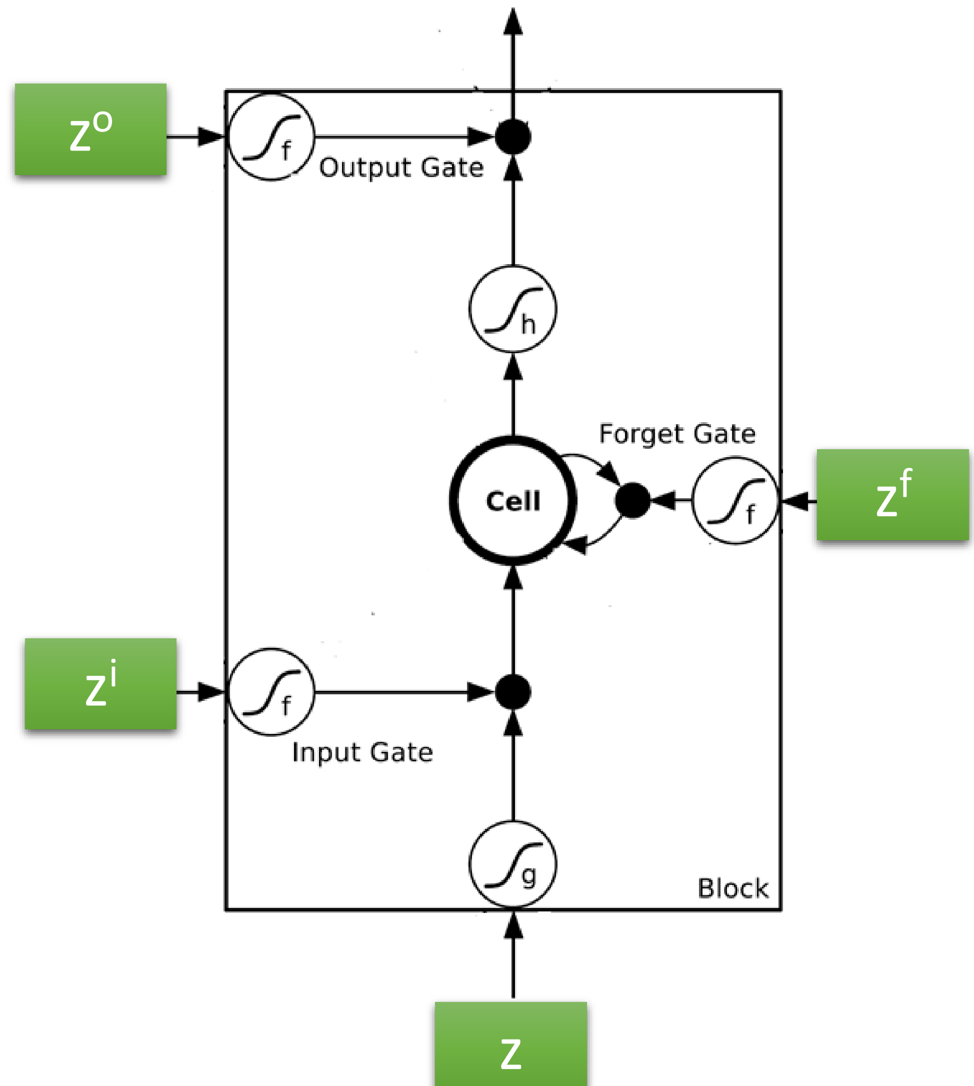


c^{t-1}
vector
Output Gate
Forget Gate
Cell
Input Gate
Block
Output Gate
Forget Gate
Cell
Input Gate
Block
……
Output Gate
Forget Gate
Cell
Input Gate
Block
z^f
z^i
z
z^o
4 vectors
x^t

LSTM



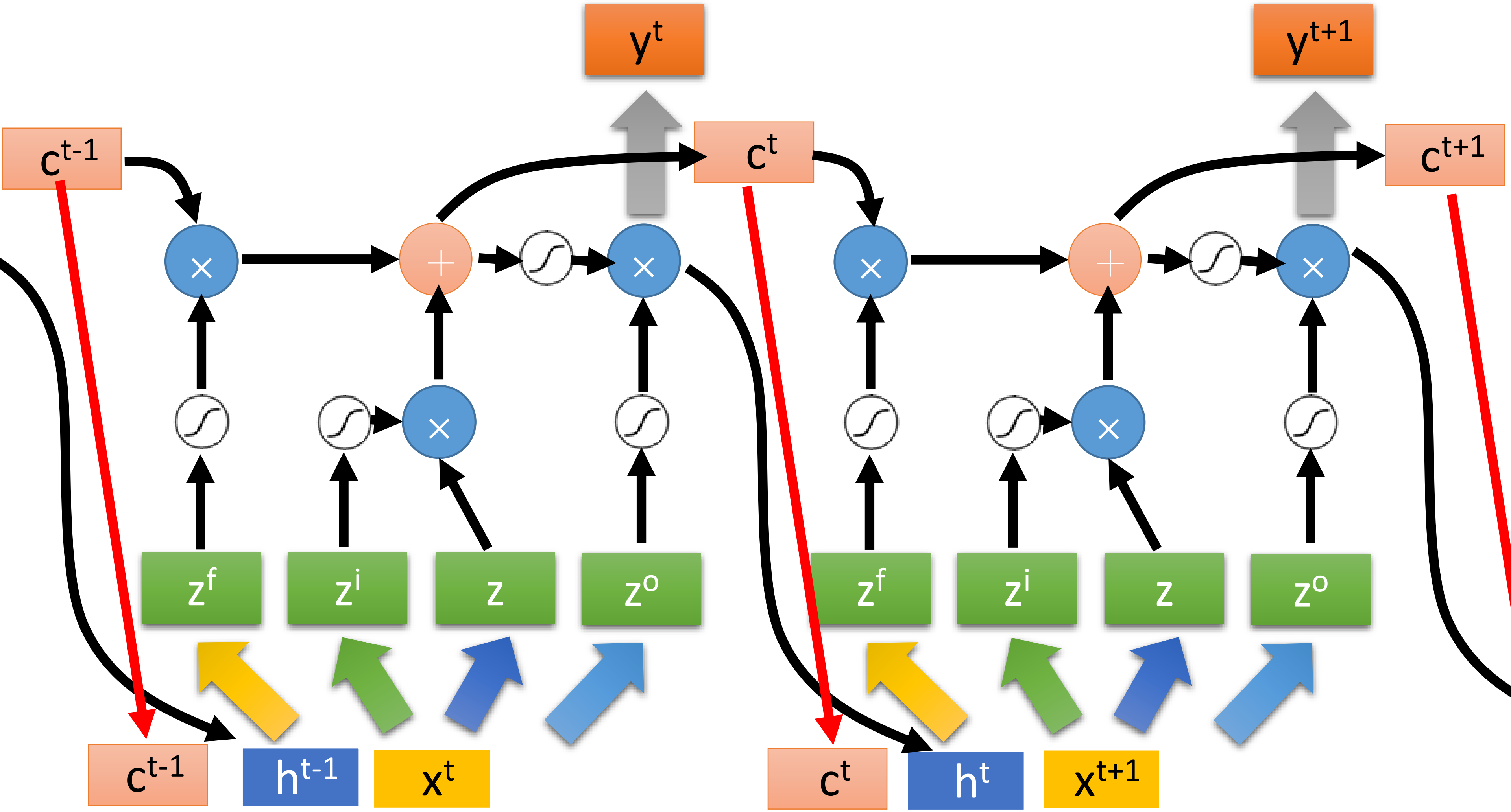
c^t-1
y^t
z^f
z^i
z
z^o
x^t



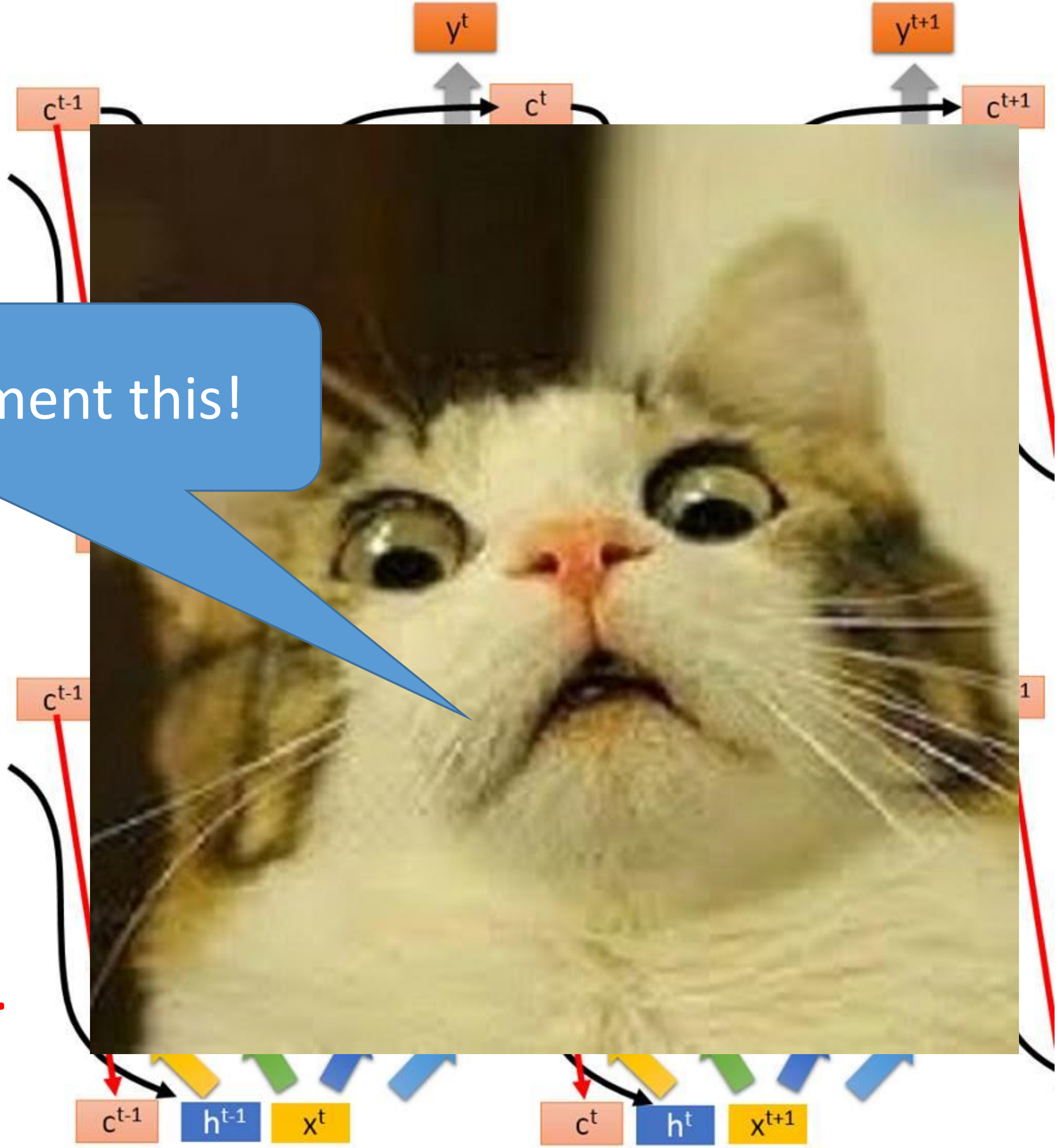
z^o
Output Gate
f
h
Forget Gate
Cell
f
z^f
z^i
f
Input Gate
g
Block
z

LSTM

Extension: “peephole”



Multiple-layer LSTM



I will not implement this!

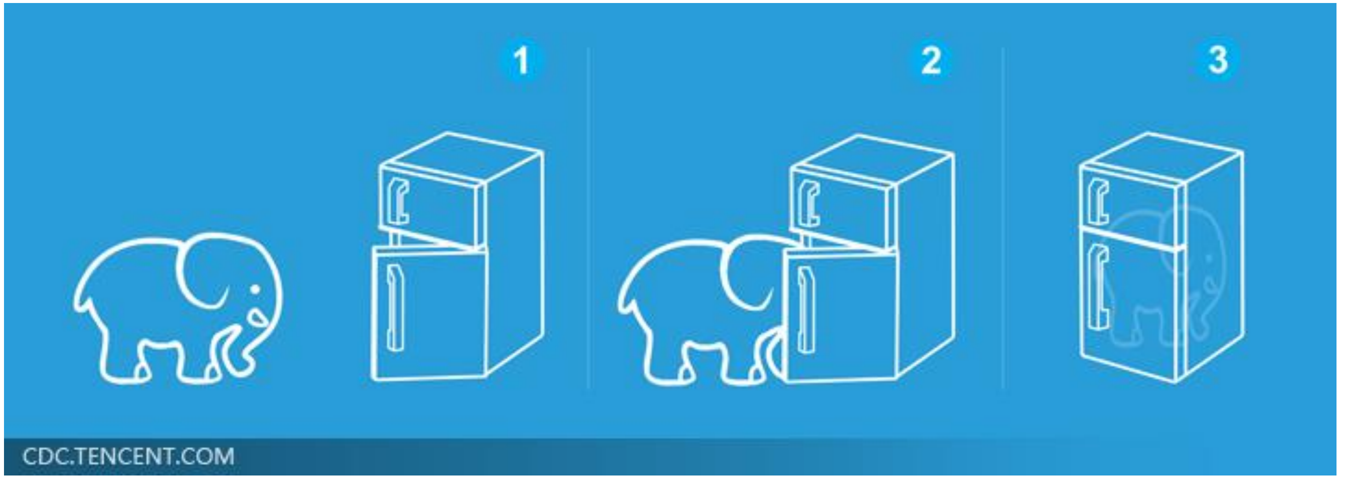
This is quite standard now ...

https://img.komicolle.org/2015-09-20/src/14426967627131.gif

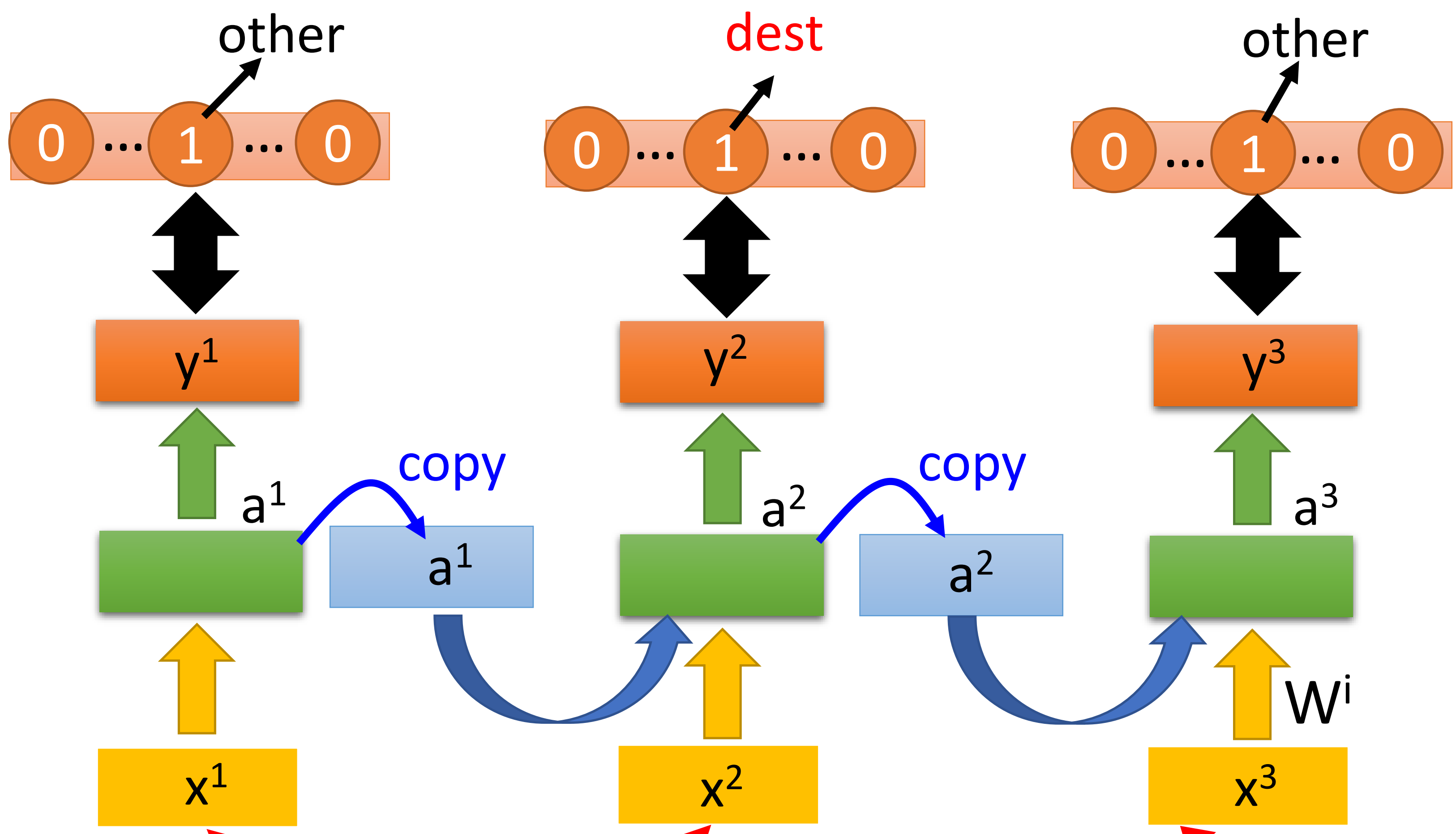
Three Steps for Deep Learning

Step 1: define a set of function → **Step 2: goodness of function** → Step 3: pick the best function

Deep Learning is so simple ……



Learning Target



Training Sentences:

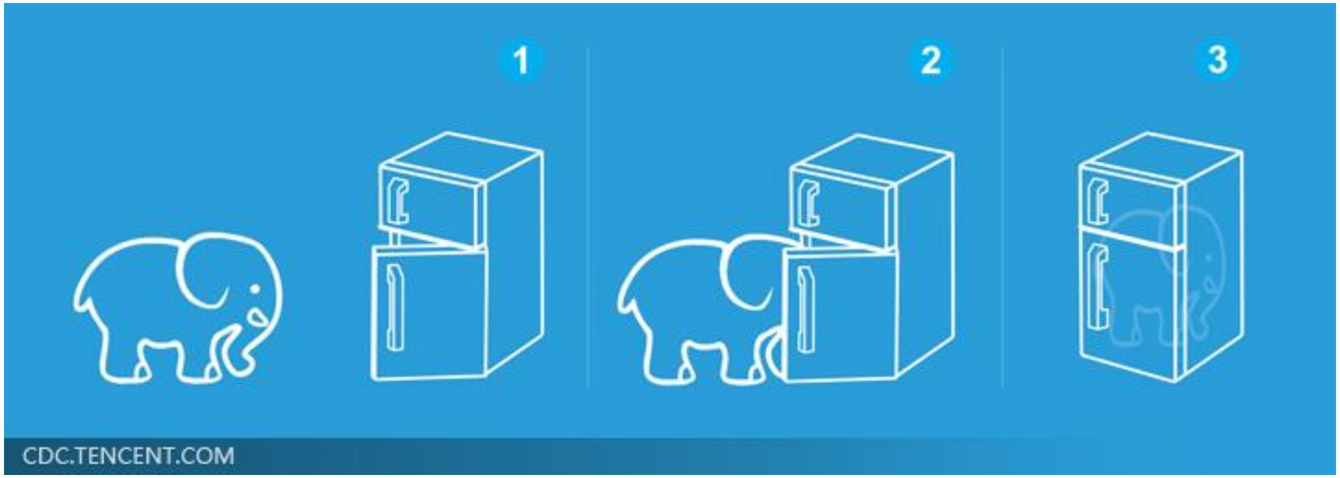
arrive Taipei on November 2nd

other dest other time time

Three Steps for Deep Learning

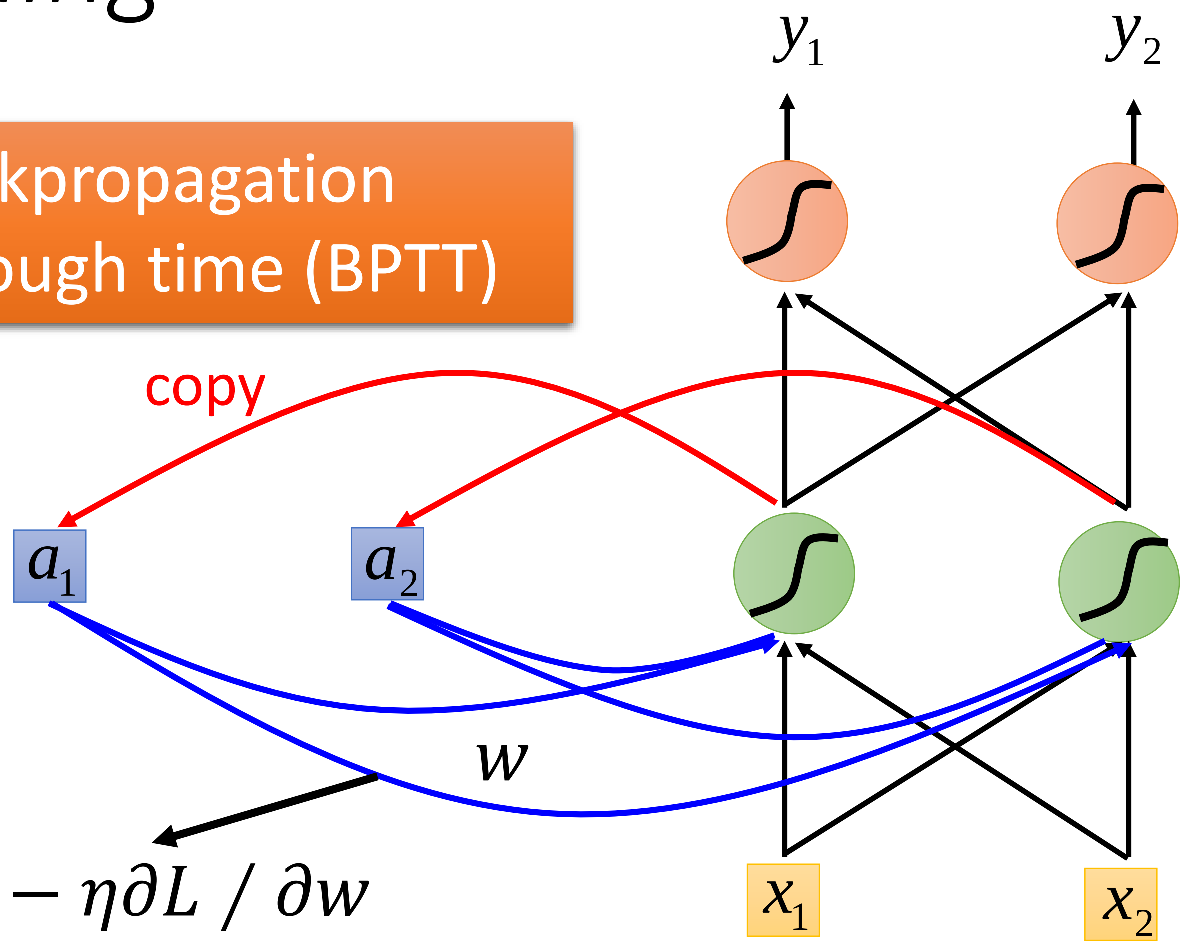
Step 1: define a set of function → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……



Learning

Backpropagation through time (BPTT)

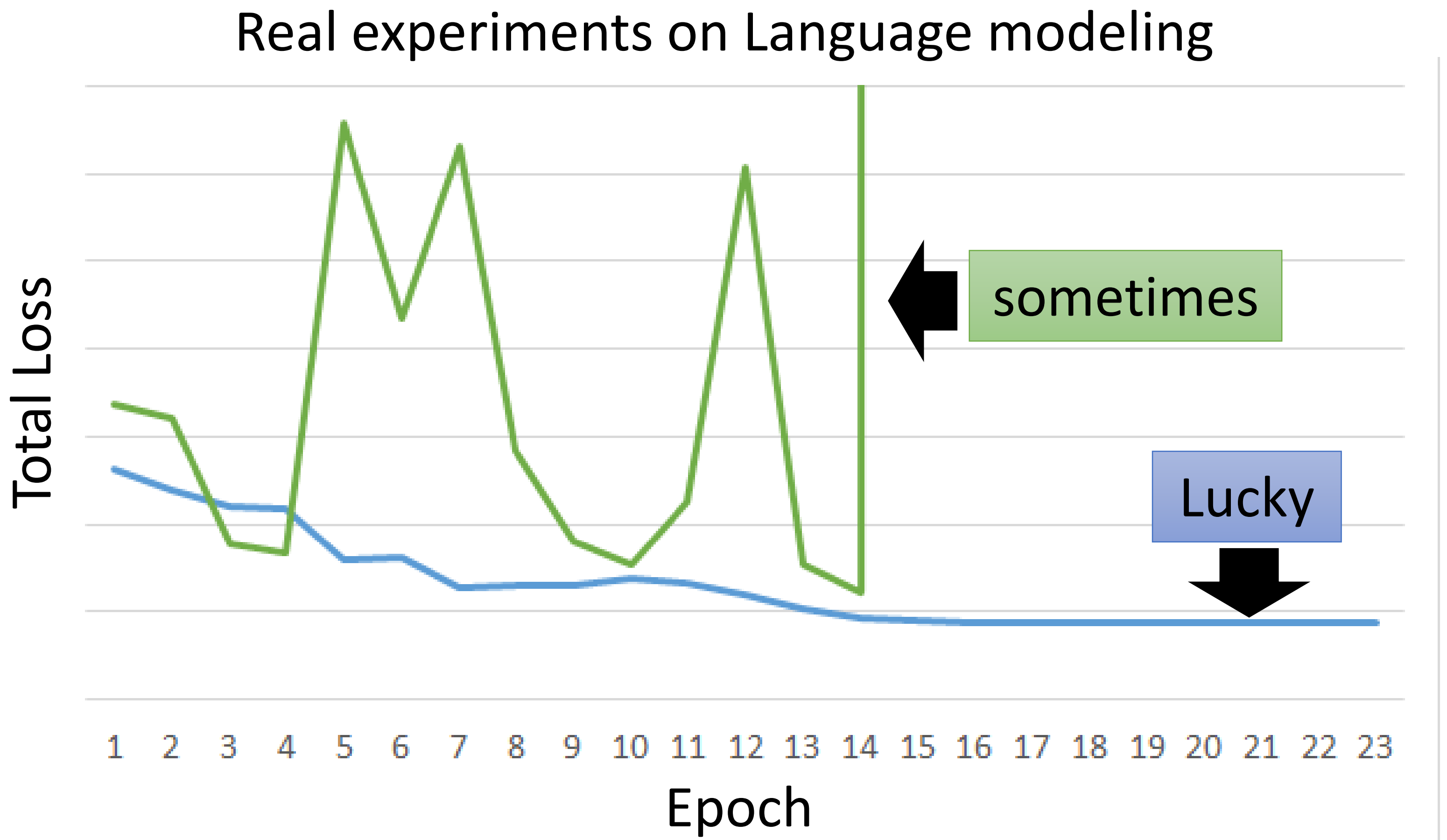


$$w \leftarrow w - \eta \partial L / \partial w$$

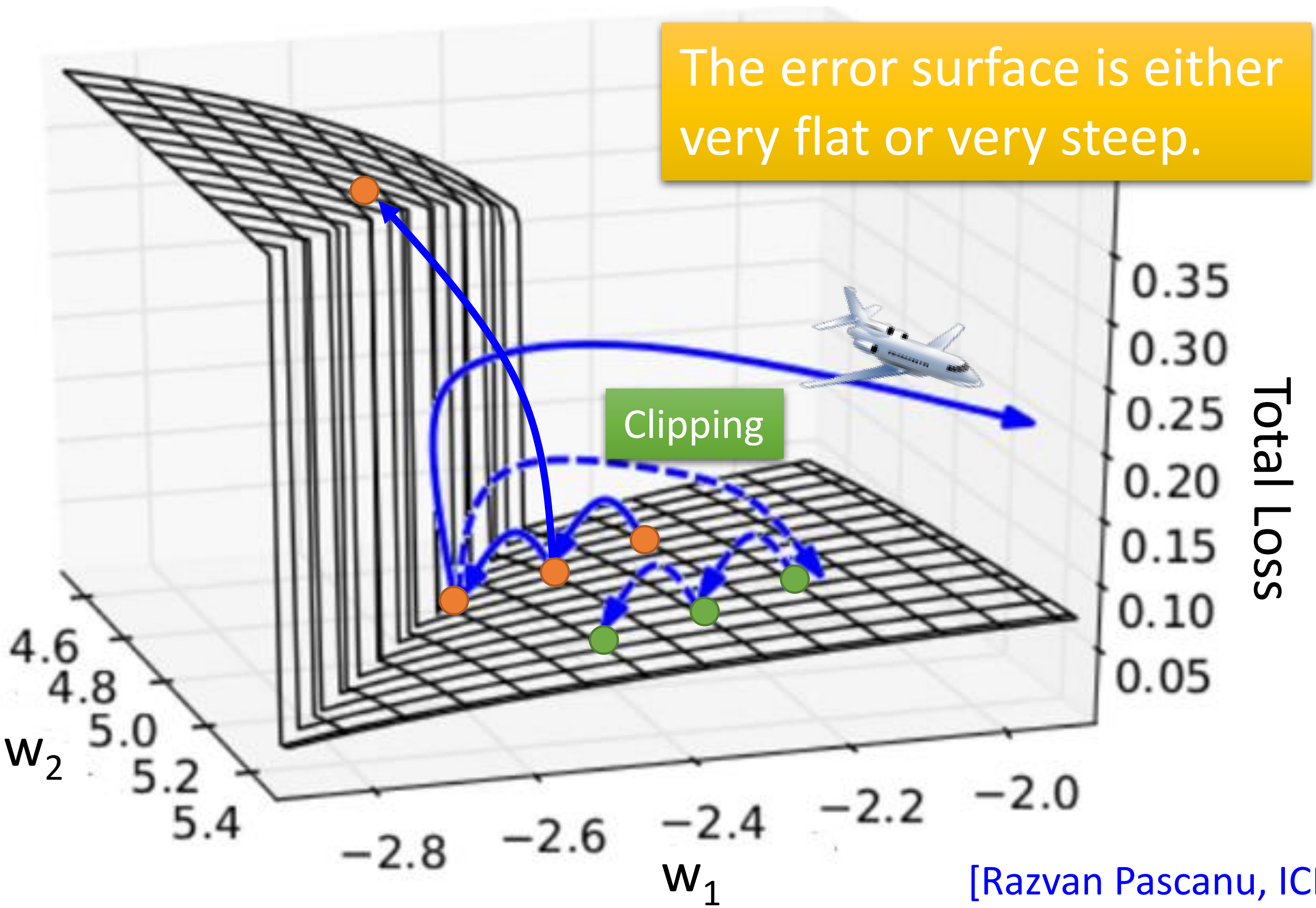
RNN Learning is difficult in practice.

Unfortunately ……

- RNN-based network is not always easy to learn



The error surface is rough.



[Razvan Pascanu, ICML'13]

Why?

$w = 0.01 \Rightarrow y^{1000} \approx 0$

$w = 0.99 \Rightarrow y^{1000} \approx 0$

small $\partial L/\partial w$ ➡ Large Learning rate?

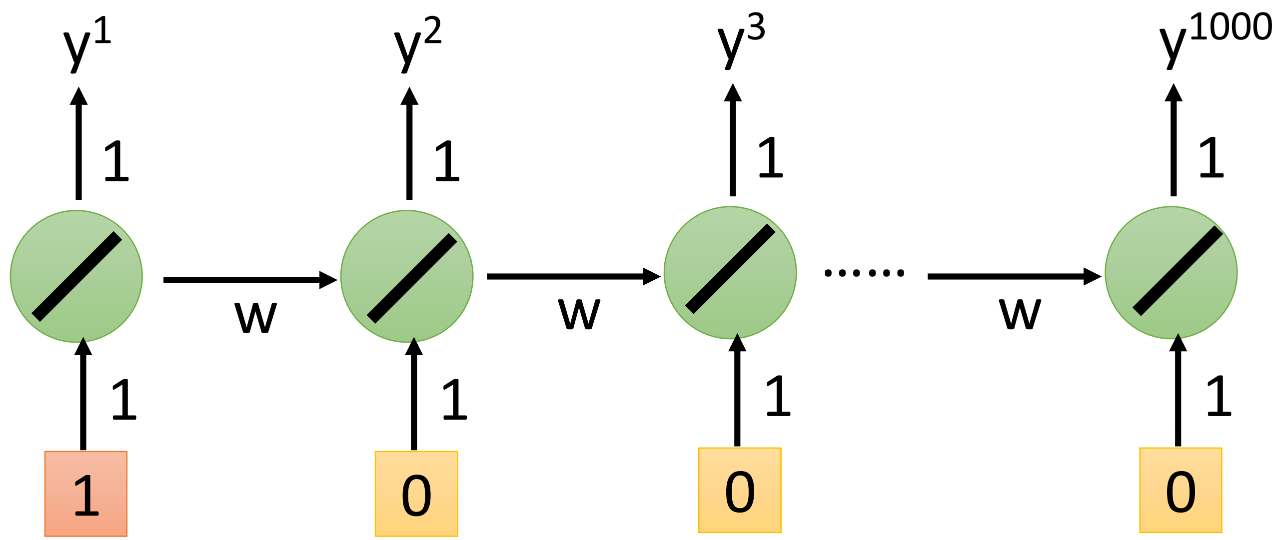
$w = 1 \Rightarrow y^{1000} = 1$

$w = 1.01 \Rightarrow y^{1000} \approx 20000$

Large $\partial L/\partial w$ ➡ Small Learning rate?

$=w^{999}$

Toy Example



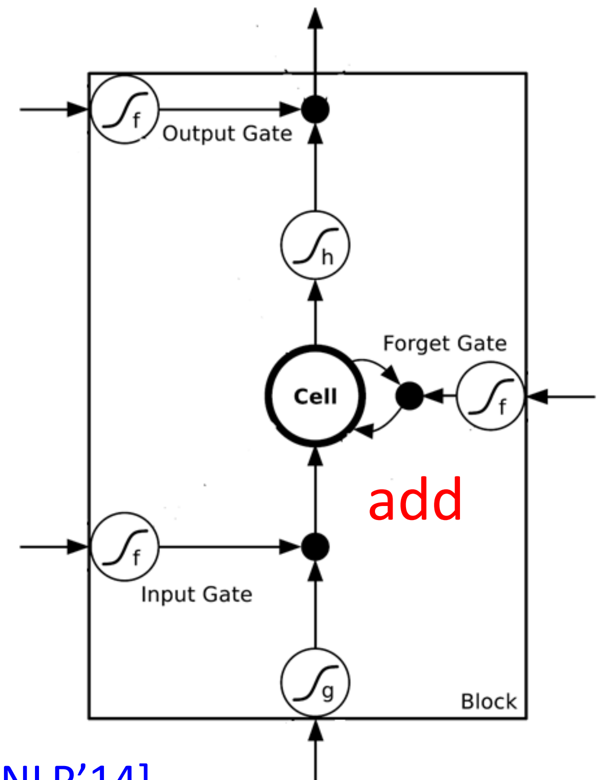
Helpful Techniques

- Long Short-term Memory (LSTM)
  - Can deal with gradient vanishing (not gradient explode)

➢ Memory and input are ***added***

➢ The influence never disappears unless forget gate is closed

No Gradient vanishing
(If forget gate is opened.)

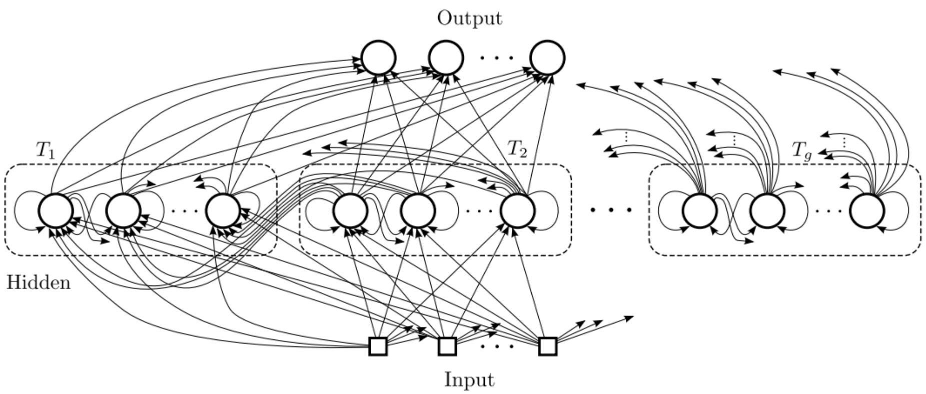


Gated Recurrent Unit (GRU): simpler than LSTM

[Cho, EMNLP'14]

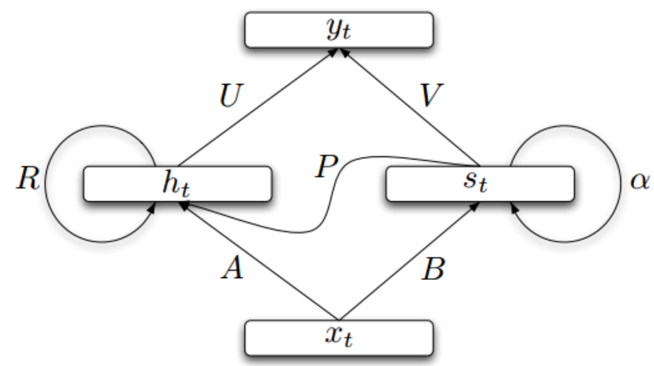
Helpful Techniques

Clockwise RNN



[Jan Koutnik, JMLR'14]

Structurally Constrained Recurrent Network (SCRN)

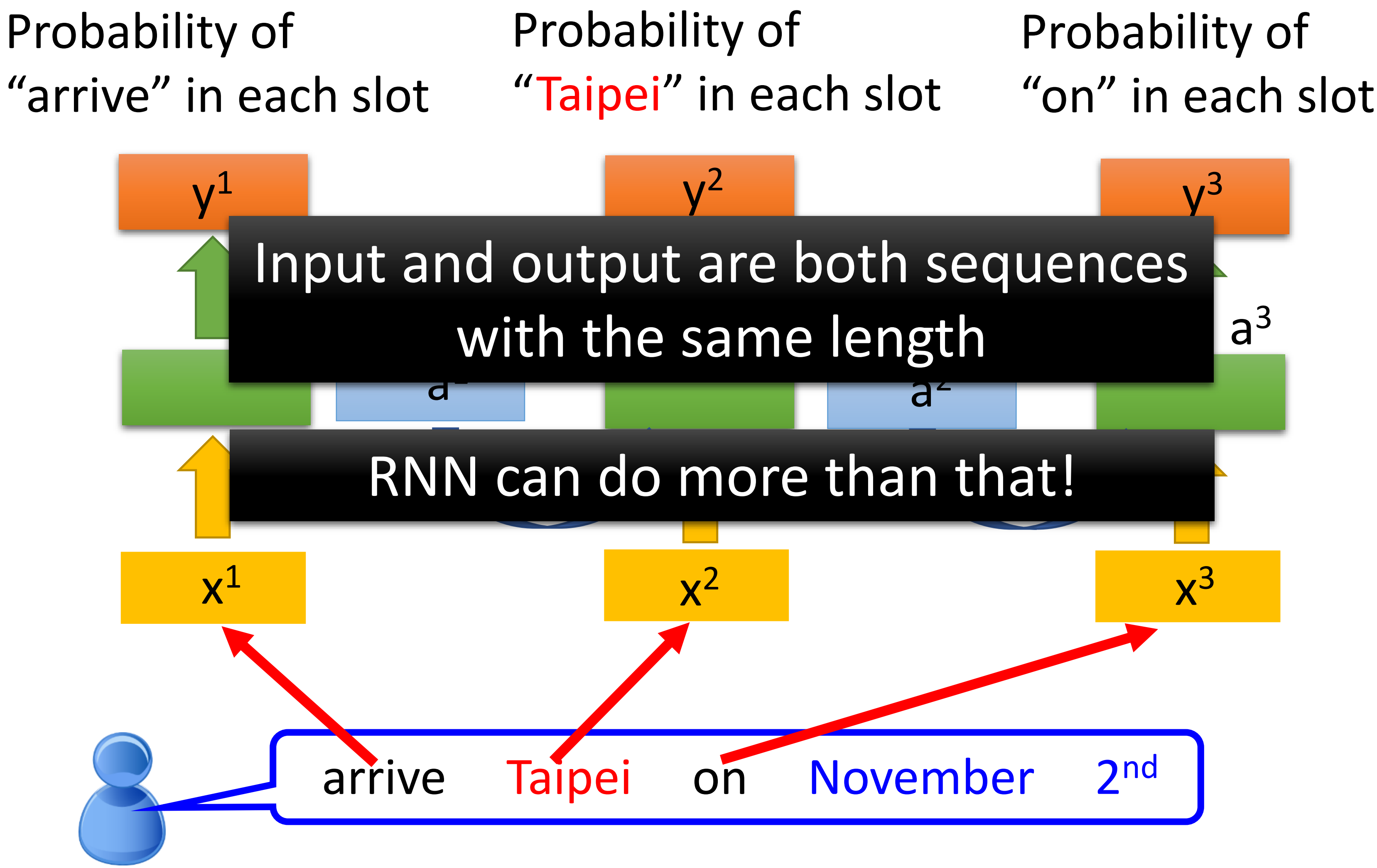


[Tomas Mikolov, ICLR'15]

Vanilla RNN Initialized with Identity matrix + ReLU activation function [Quoc V. Le, arXiv'15]

- Outperform or be comparable with LSTM in 4 different tasks

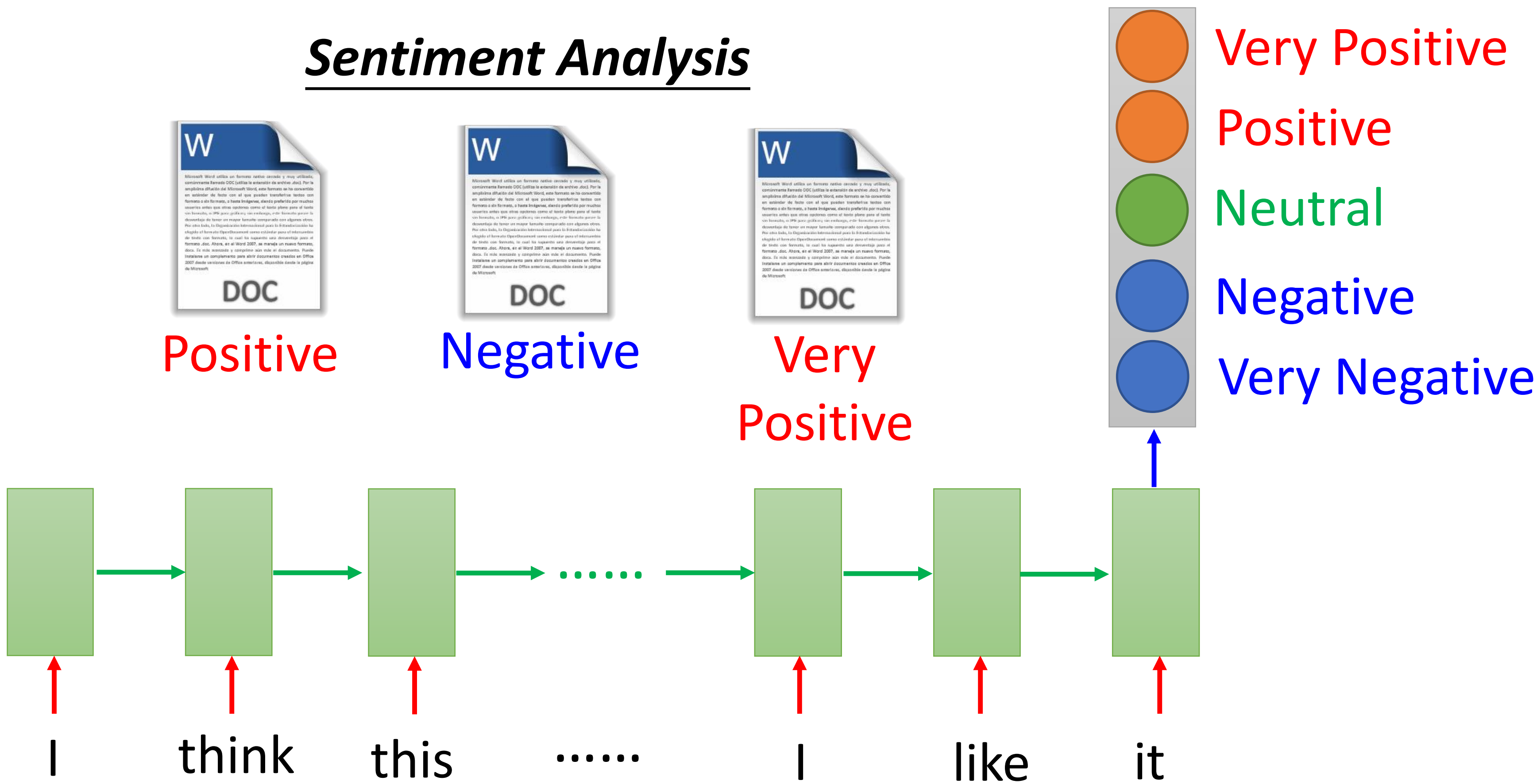
More Applications ……



Many to one

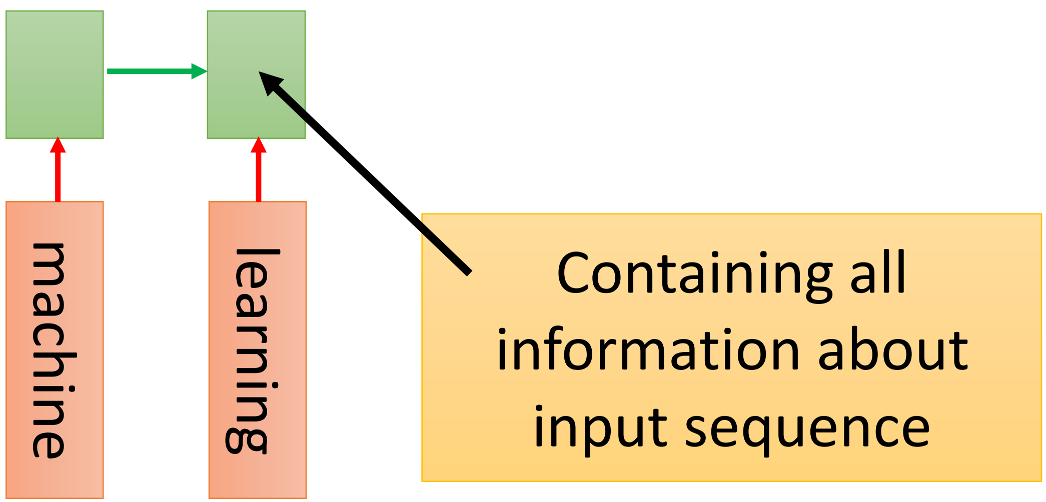
Keras Example: https://github.com/fchollet/keras/blob/master/examples/imdb_lstm.py

- Input is a ***vector sequence***, but output is only ***one vector***



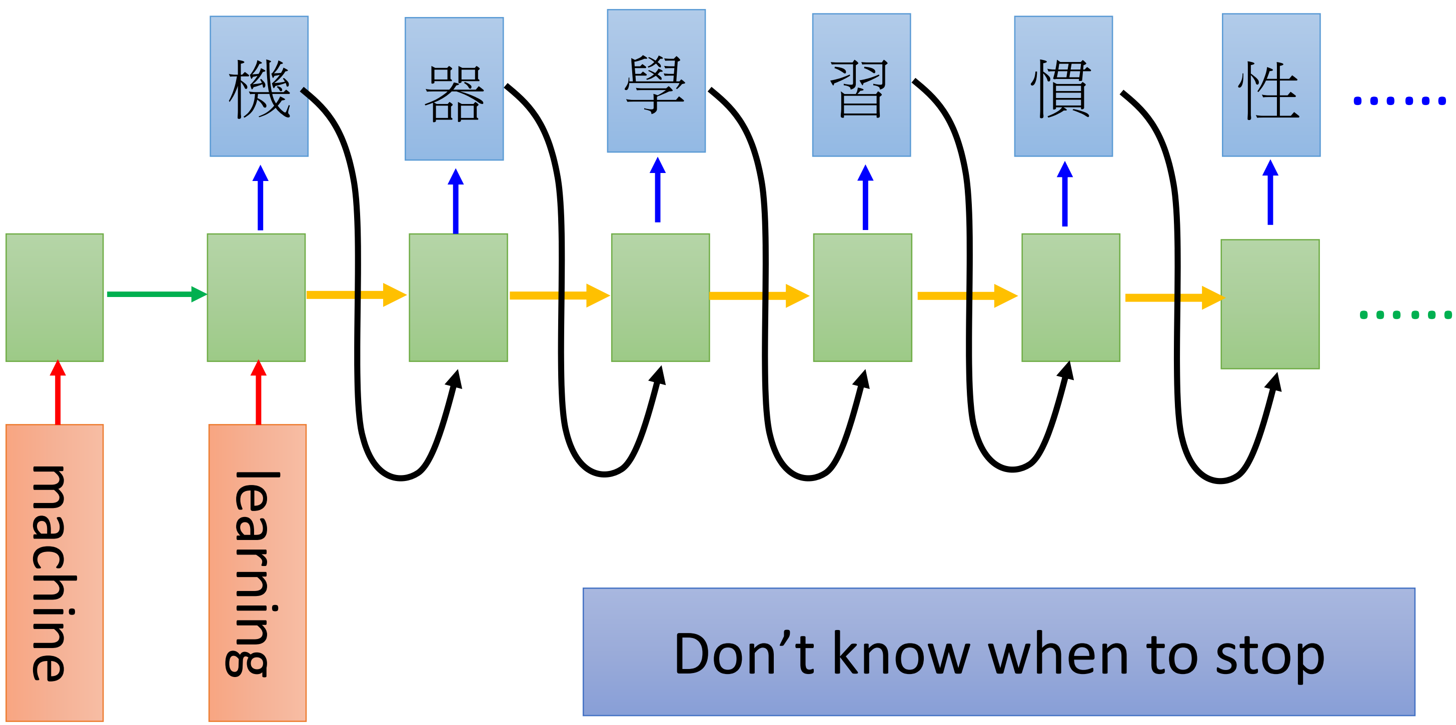
Many to Many

- Both input and output are both sequences ***with different lengths***. → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→機器學習)



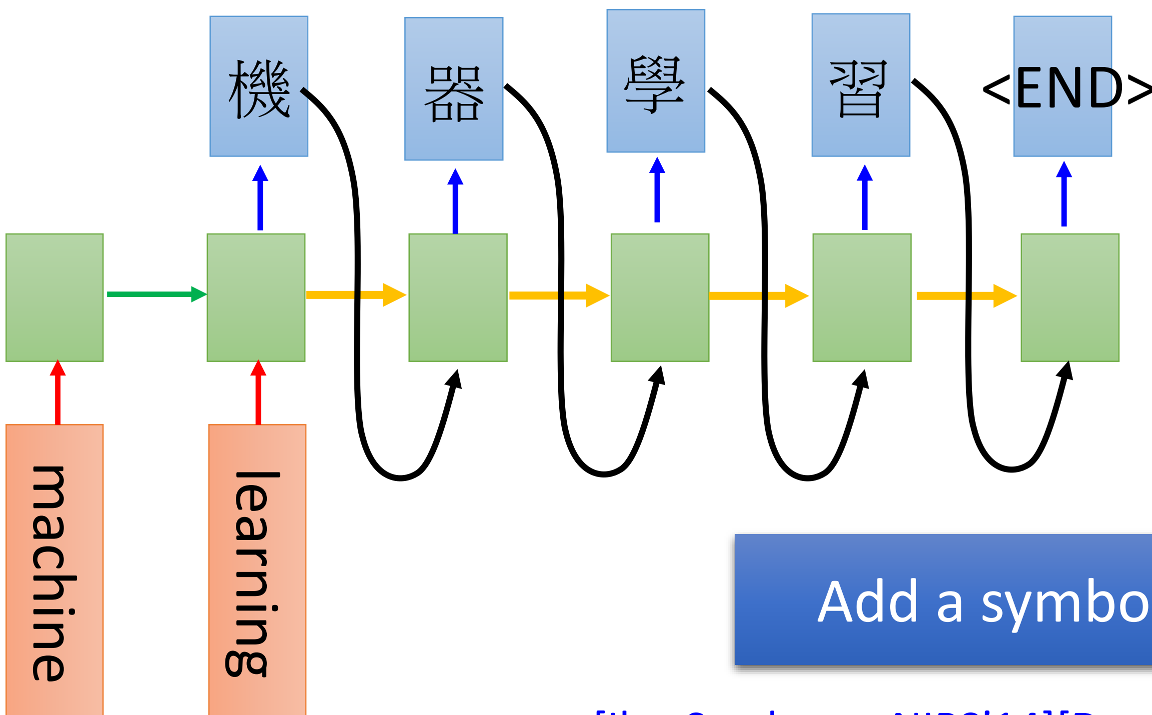
Many to Many (No Limitation)

- Both input and output are both sequences ***with different lengths***. → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→機器學習)



Many to Many (No Limitation)

- Both input and output are both sequences ***with different lengths***. → ***Sequence to sequence learning***
  - E.g. ***Machine Translation*** (machine learning→機器學習)

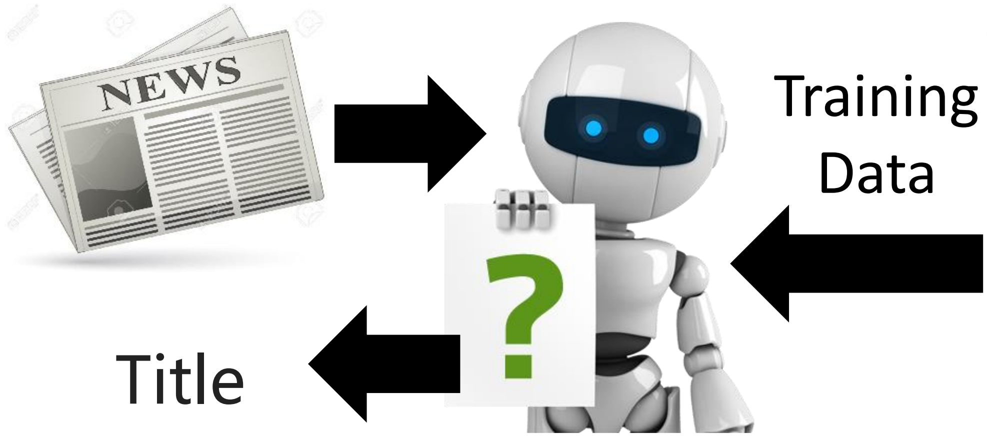


Add a symbol “**<END>**“

[Ilya Sutskever, NIPS’14][Dzmitry Bahdanau, arXiv’15]

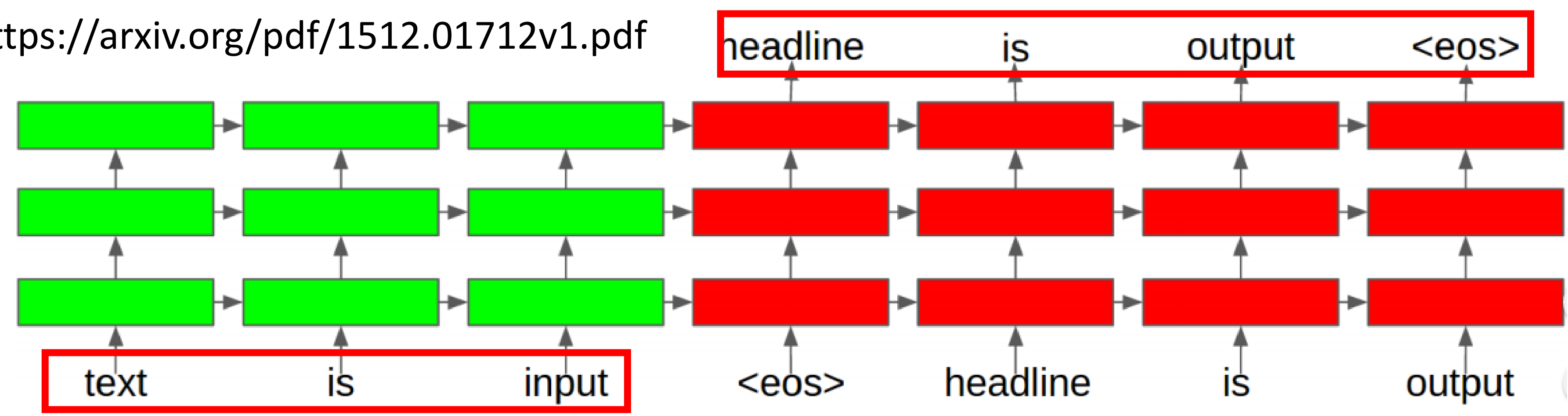
Many to Many: Title Generation

[Alexander M Rush, EMNLP 15][Chopra, NAACL 16][Lopyrev, arXiv 2015][Shen, arXiv 2016][Yu & Lee,SLT 2016]



Input: a document (word sequence), output: its title (shorter word sequence)

https://arxiv.org/pdf/1512.01712v1.pdf

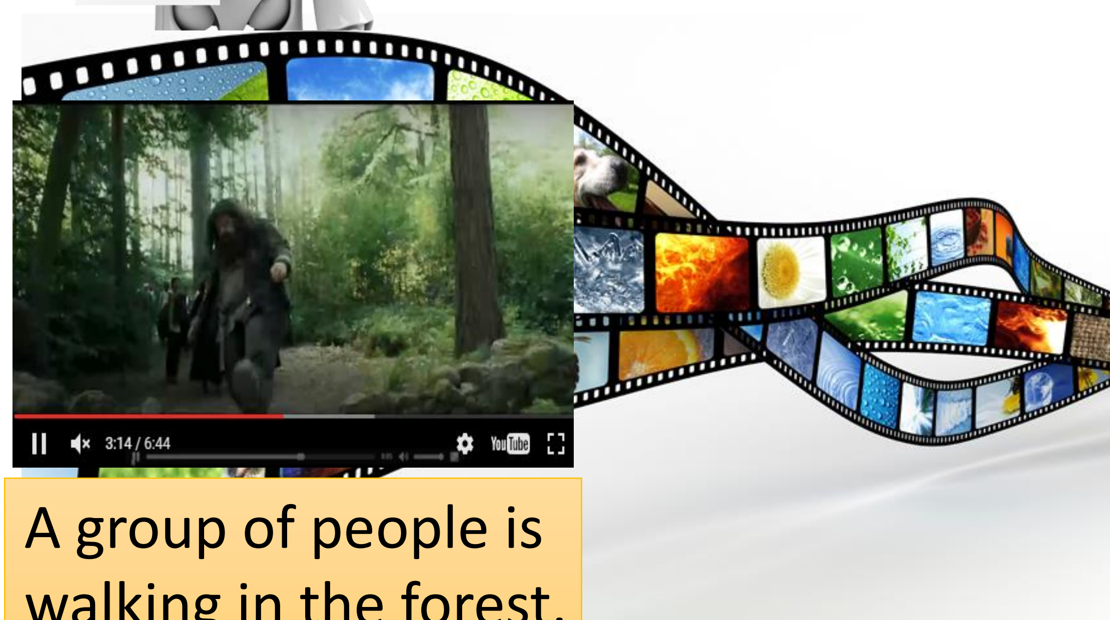


Many to Many:
Video Caption Generation

Video

A girl is running.

A group of people is knocked by a tree.



A group of people is walking in the forest.

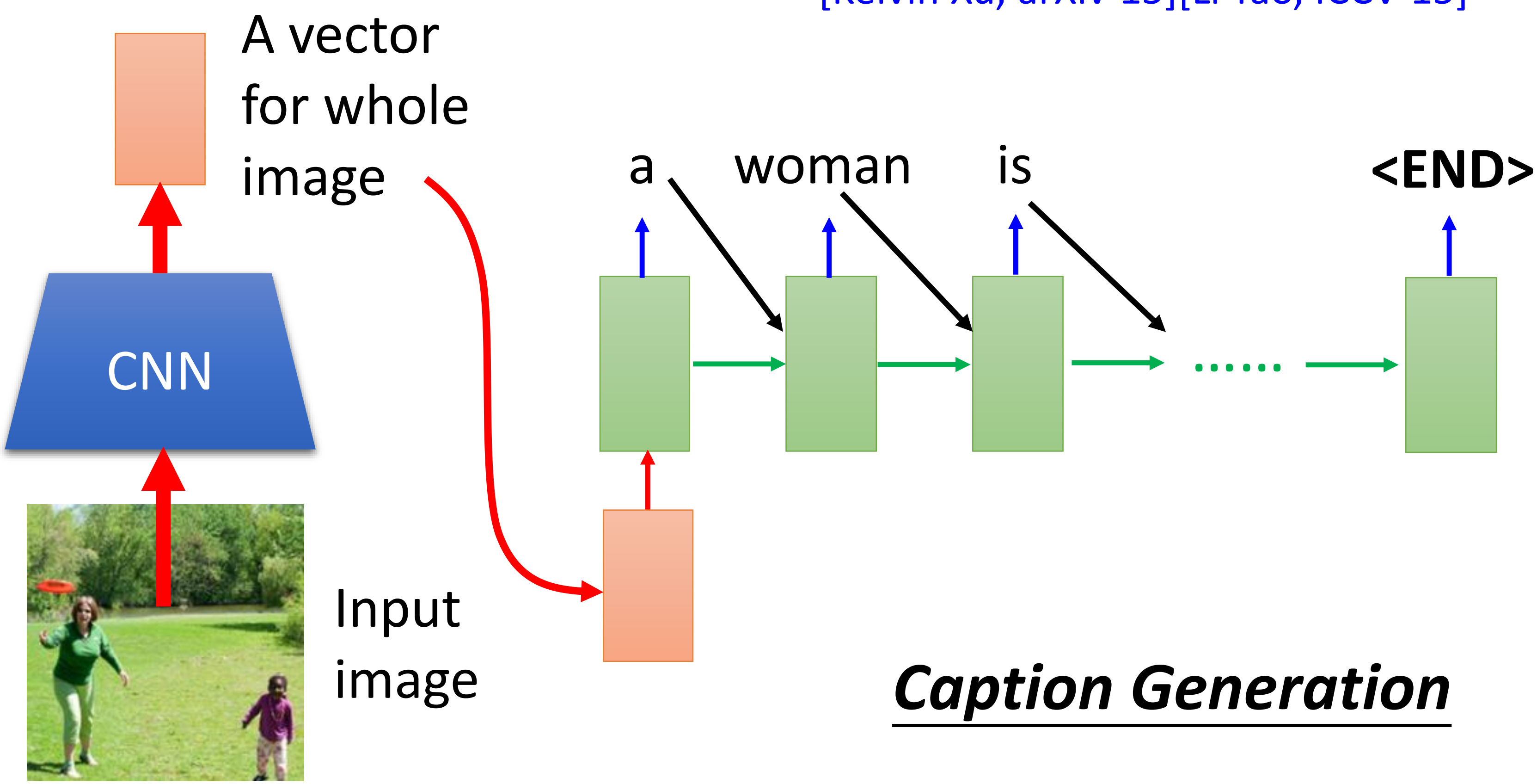
Many to Many:
Video Caption Generation

- Can machine describe what it see from video?

One to Many:
Image Caption Generation

- Input an image, but output a sequence of words

[Kelvin Xu, arXiv'15][Li Yao, ICCV'15]

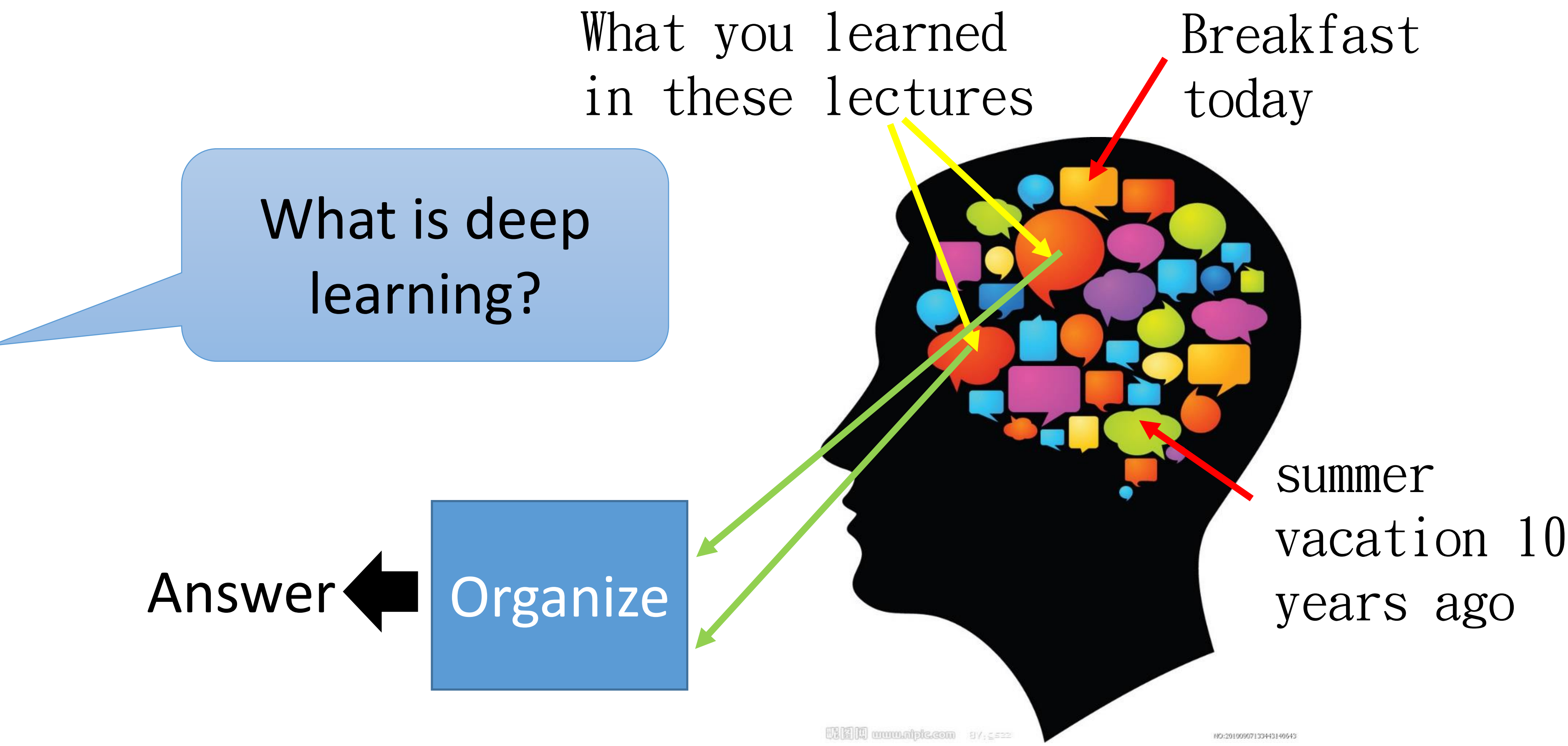


One to Many: Image Caption Generation

- Can machine describe what it see from image?

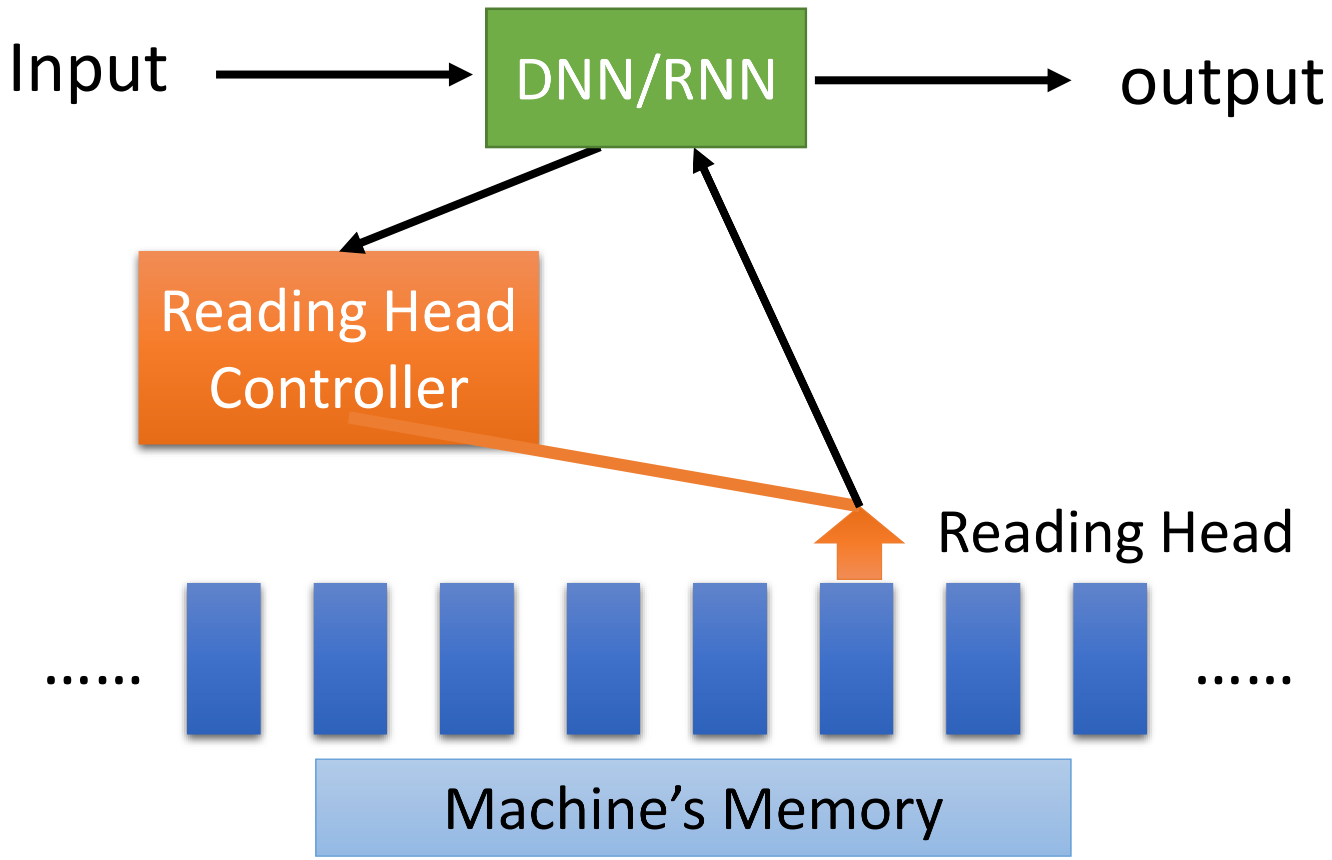
Project of MTK

Attention-based Model



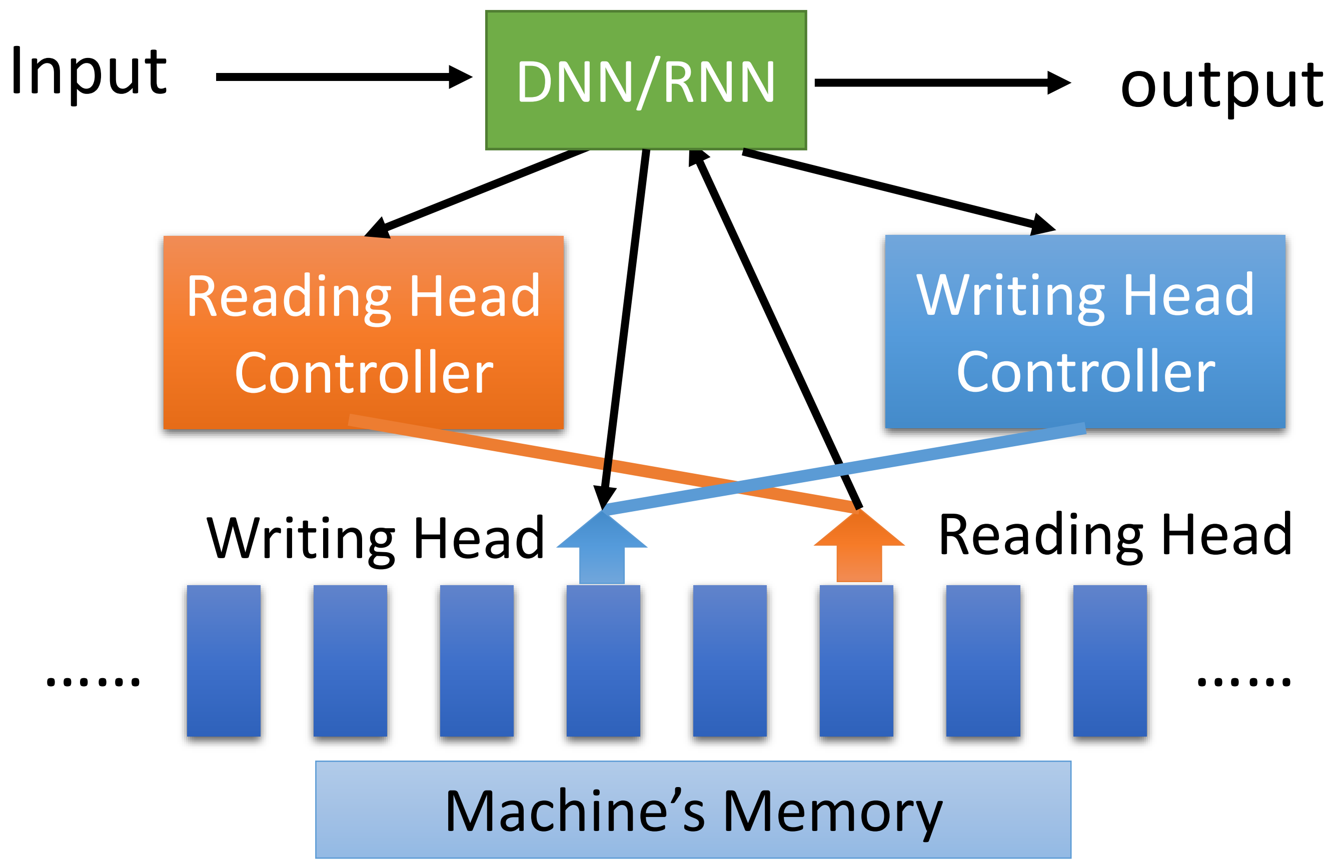
http://henrylo1605.blogspot.tw/2015/05/blog-post_56.html

Attention-based Model



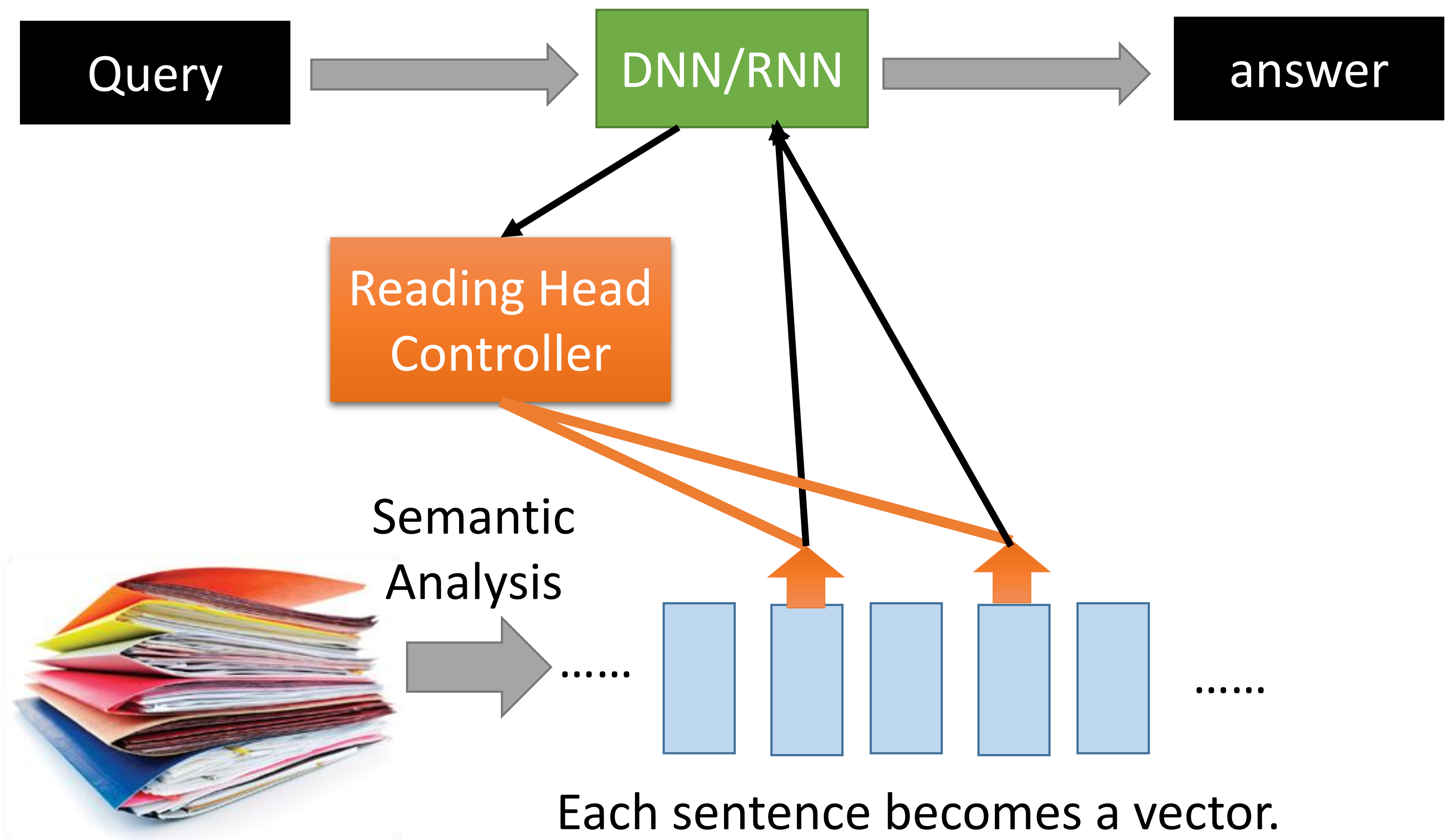
Ref:
http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Attain%20(v3).ecm.mp4/index.html

Attention-based Model v2



Neural Turing Machine

Reading Comprehension



Reading Comprehension

- End-To-End Memory Networks. S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus. NIPS, 2015.

The position of reading head:

Story (16: basic induction)	Support	Hop 1	Hop 2	Hop 3
Brian is a frog.	yes	0.00	0.98	0.00
Lily is gray.		0.07	0.00	0.00
Brian is yellow.	yes	0.07	0.00	1.00
Julius is green.		0.06	0.00	0.00
Greg is a frog.	yes	0.76	0.02	0.00
What color is Greg? Answer: yellow Prediction: yellow				

Keras has example:
https://github.com/fchollet/keras/blob/master/examples/babi_memnn.py

Visual Question Answering

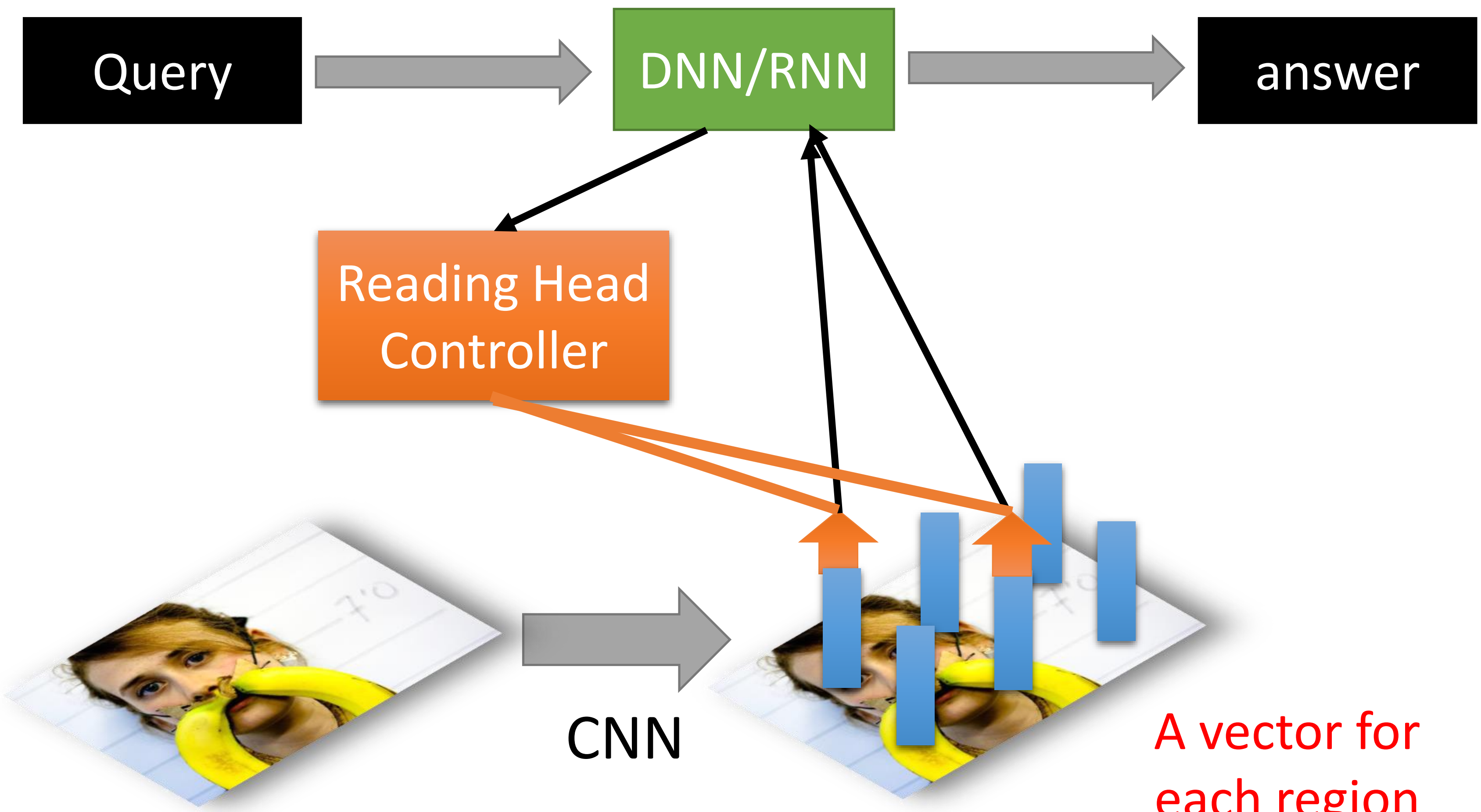
What is the mustache made of?

AI System

bananas

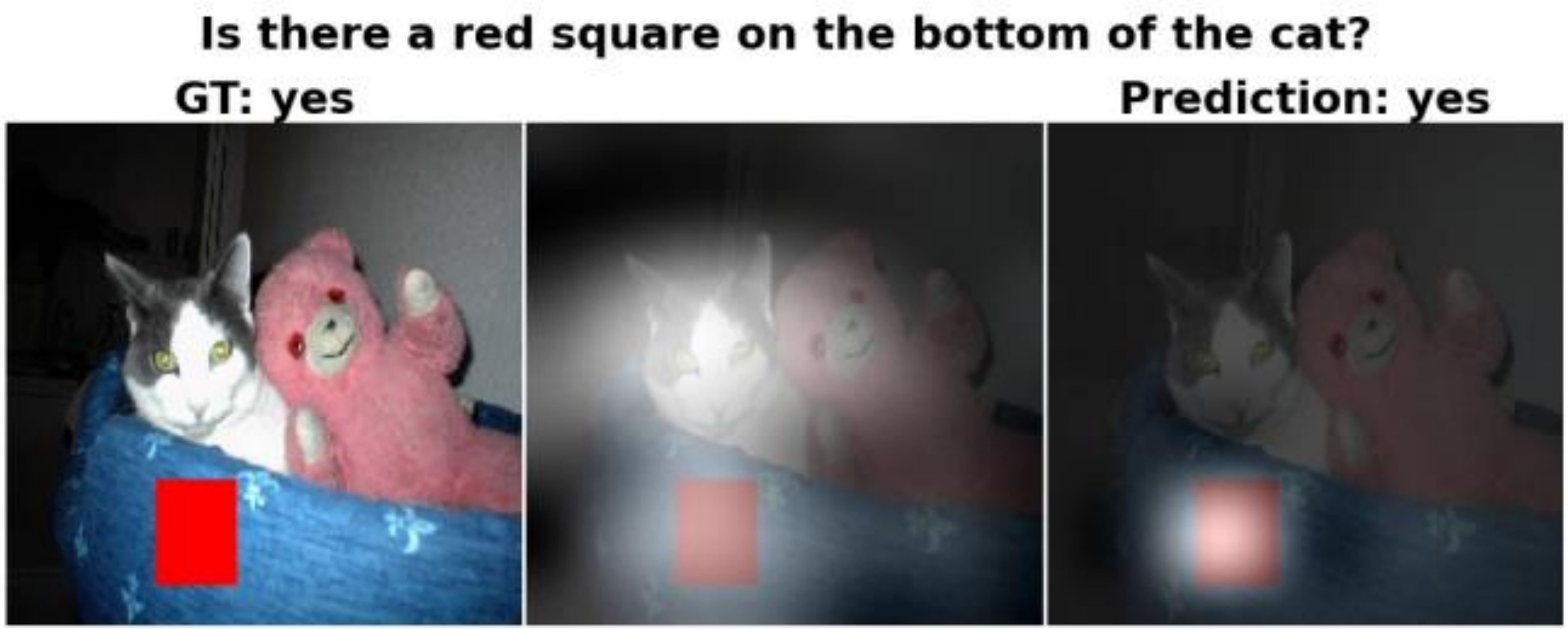
source: http://visualqa.org/

Visual Question Answering



Visual Question Answering

- Huijuan Xu, Kate Saenko. Ask, Attend and Answer: Exploring Question-Guided Spatial Attention for Visual Question Answering. arXiv Pre-Print, 2015



Speech Question Answering

- **TOEFL Listening Comprehension Test by Machine**
- Example:

Audio Story: (The original story is 5 min long.)

Question: “ What is a possible origin of Venus’ clouds? ”

Choices:

(A) gases released as a result of volcanic activity

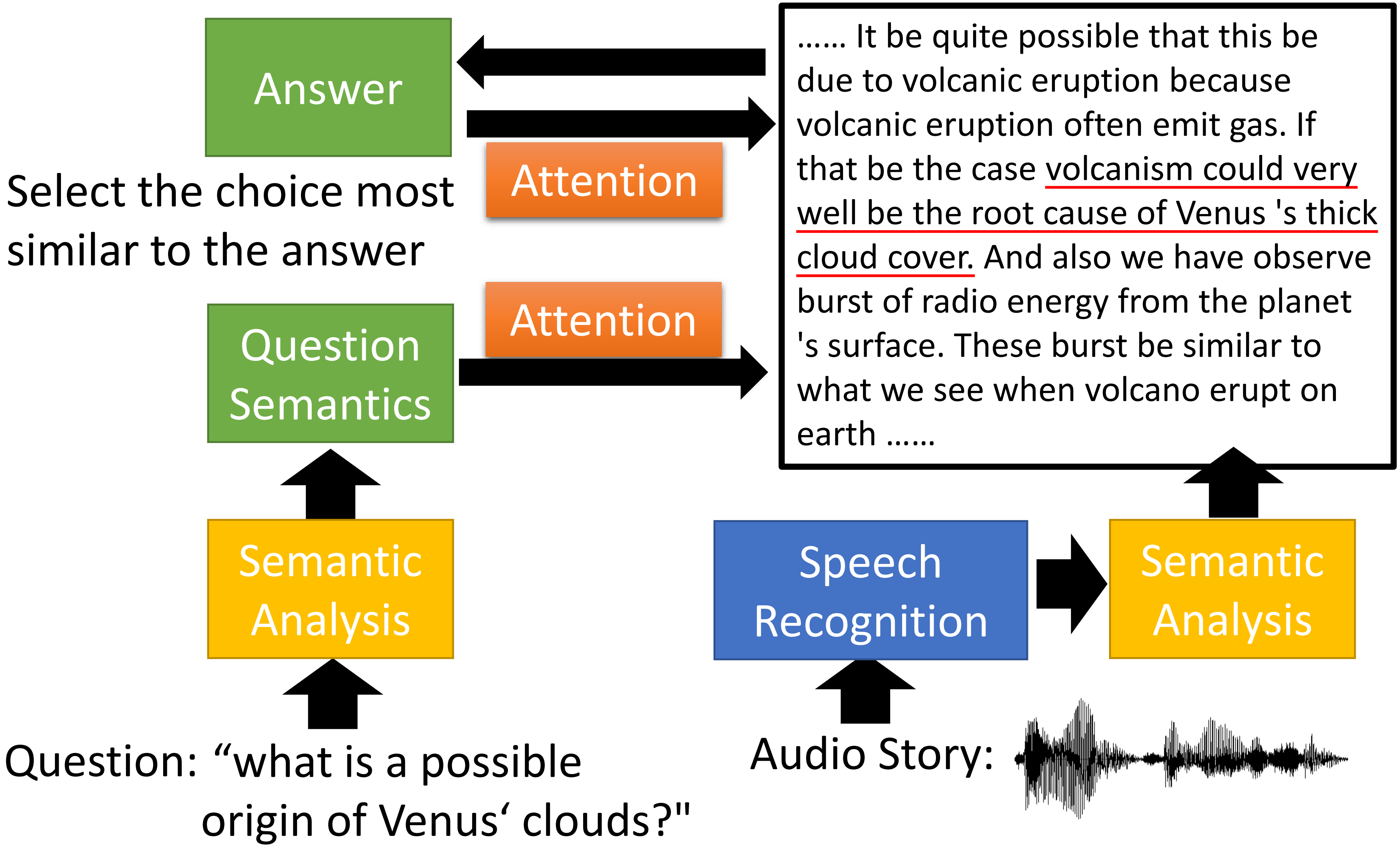
(B) chemical reactions caused by high surface temperatures

(C) bursts of radio energy from the plane's surface

(D) strong winds that blow dust into the atmosphere

Model Architecture

Everything is learned from training examples



Model Architecture - Attention Mechanism

Understand the question

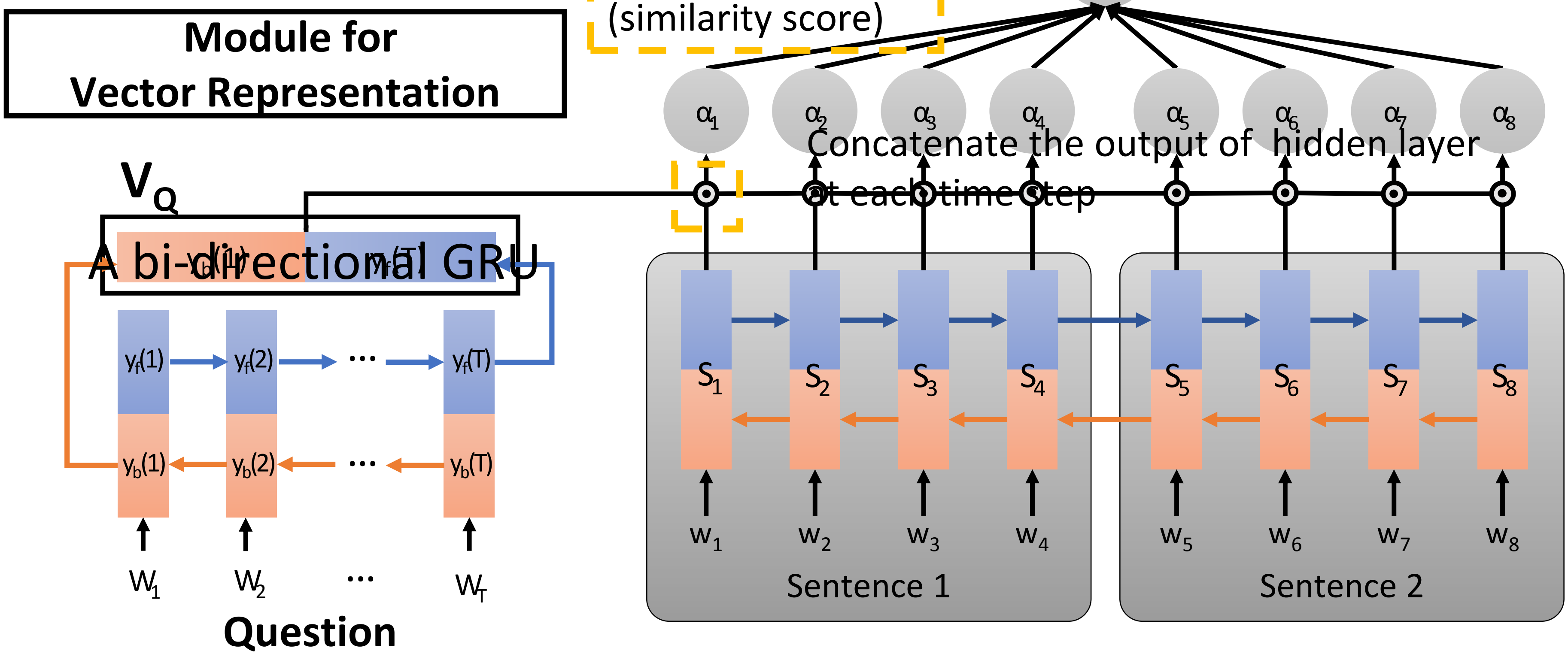
Concatenate the output of last hidden layer in bi-directional GRU

Vs : consider both Question and Story with attention weight α

$$\alpha = \frac{V_Q \cdot S_t}{|V_Q| \cdot |S_t|}$$

(similarity score)

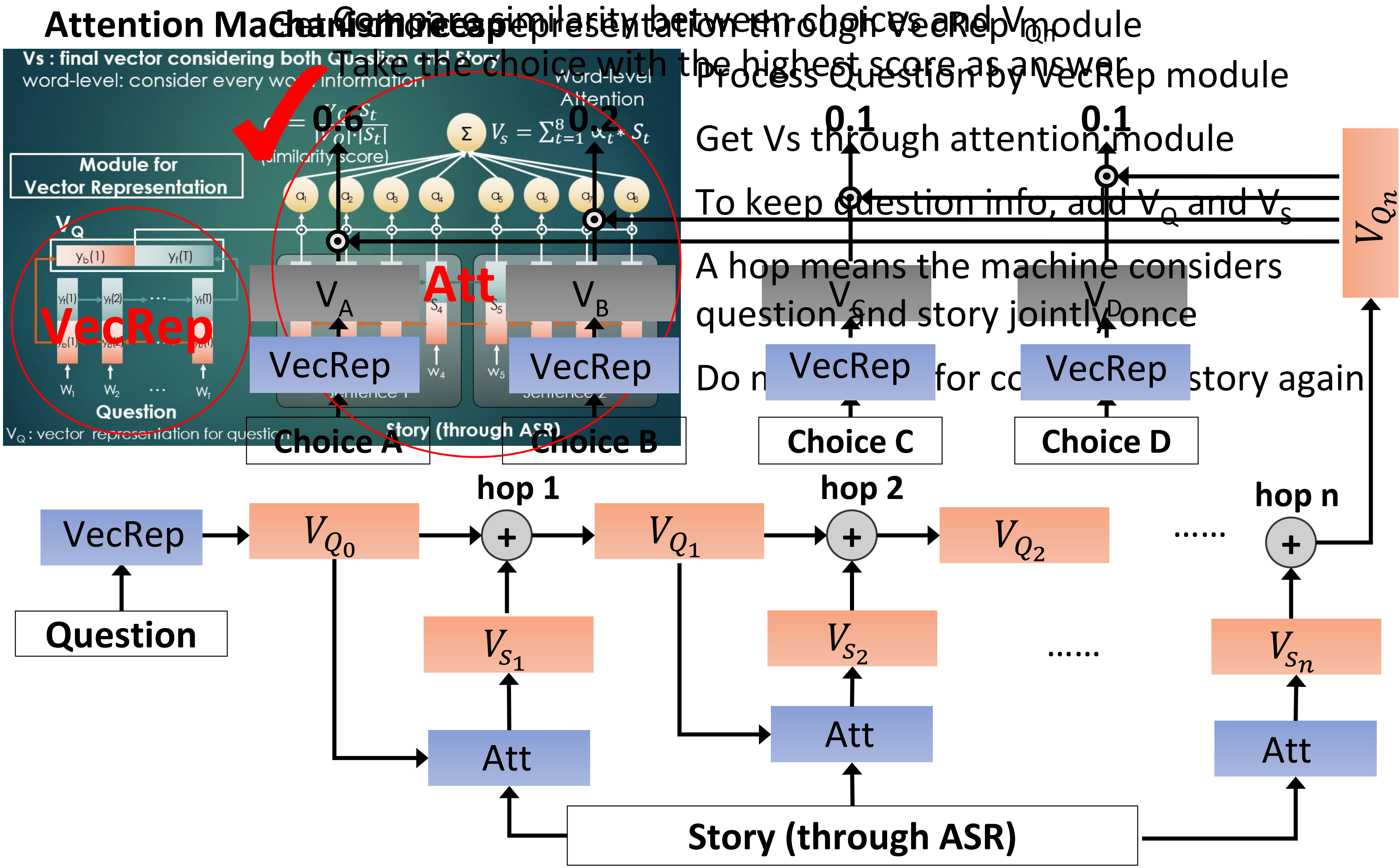
$$V_s = \sum_{t=1}^{8} \alpha_t * S_t$$



Story (through ASR)

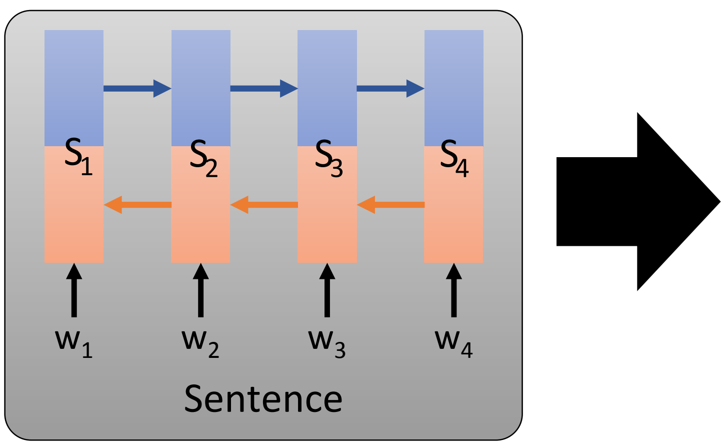
V_Q : vector representation for question

Model Architecture

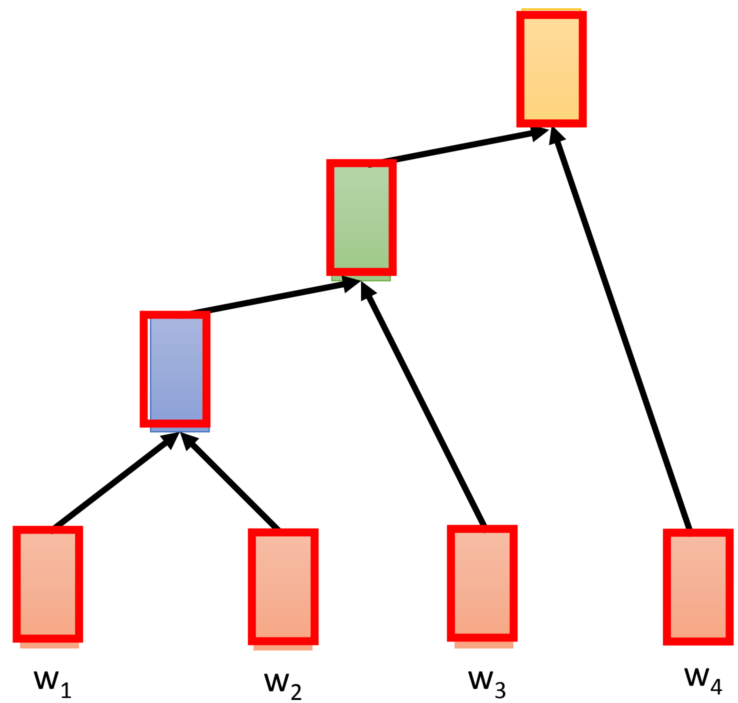


Sentence Representation

Bi-directional RNN



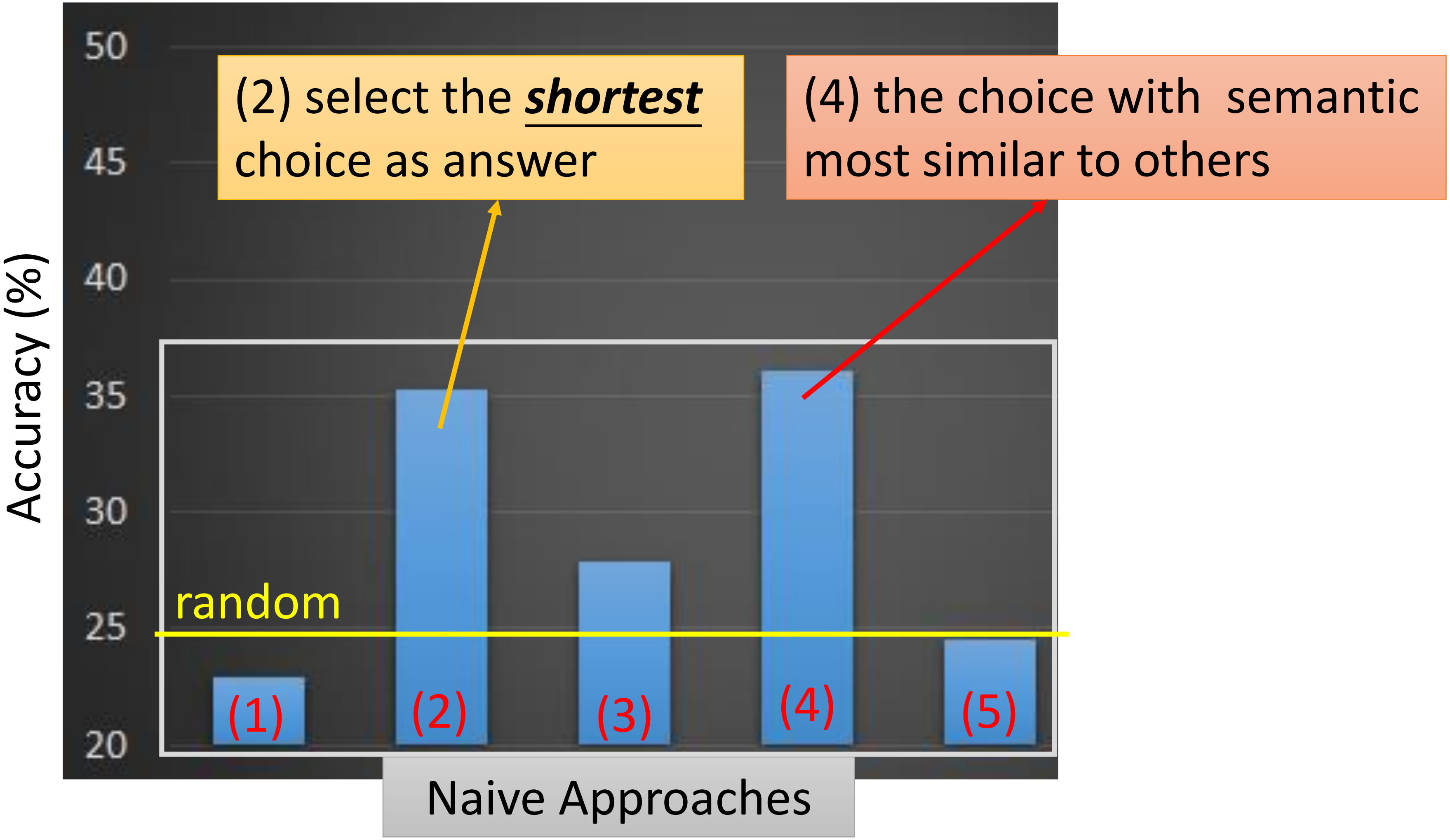
Tree-structured Neural Network



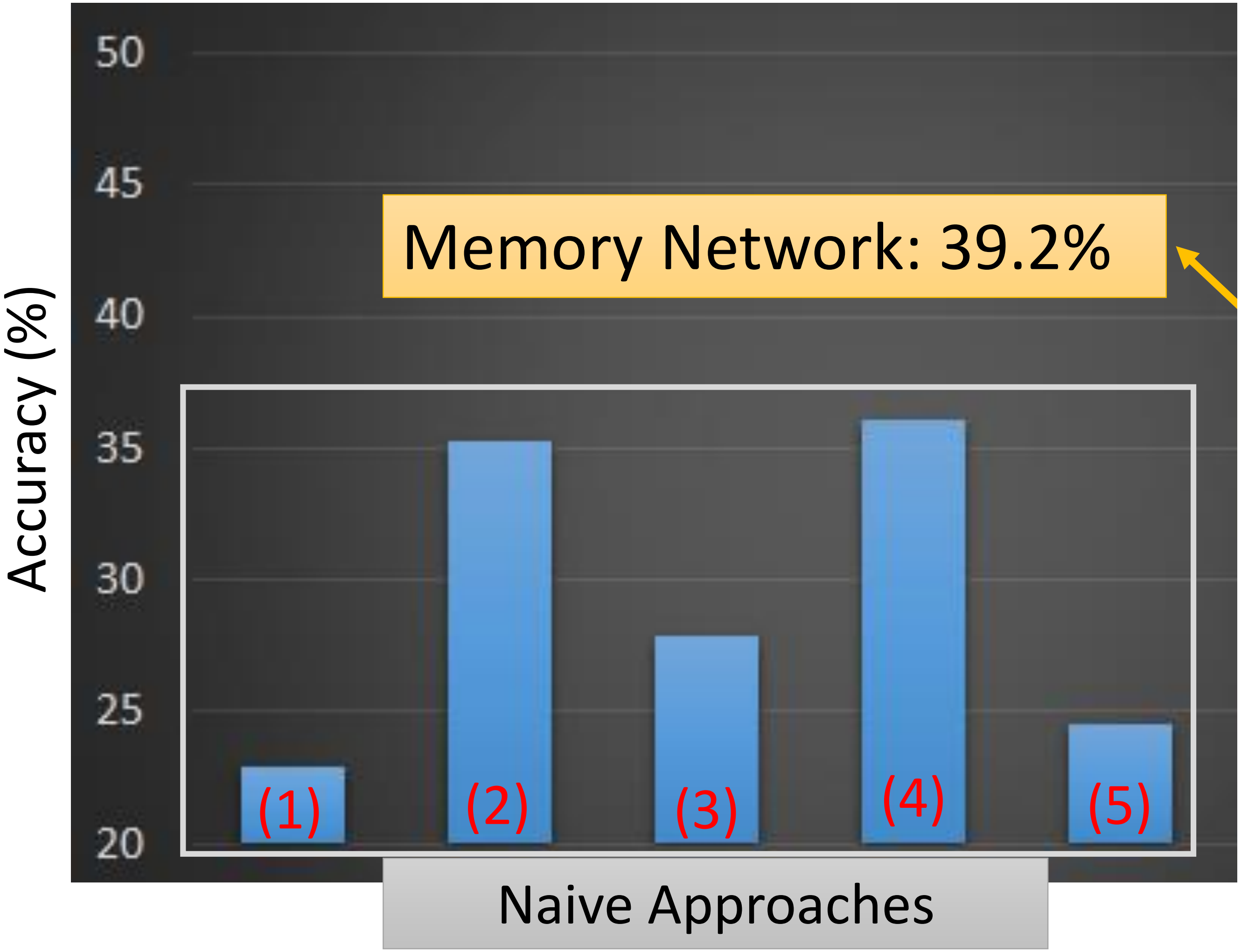
Attention on all phrases

Simple Baselines

Experimental setup:
717 for training,
124 for validation, 122 for testing

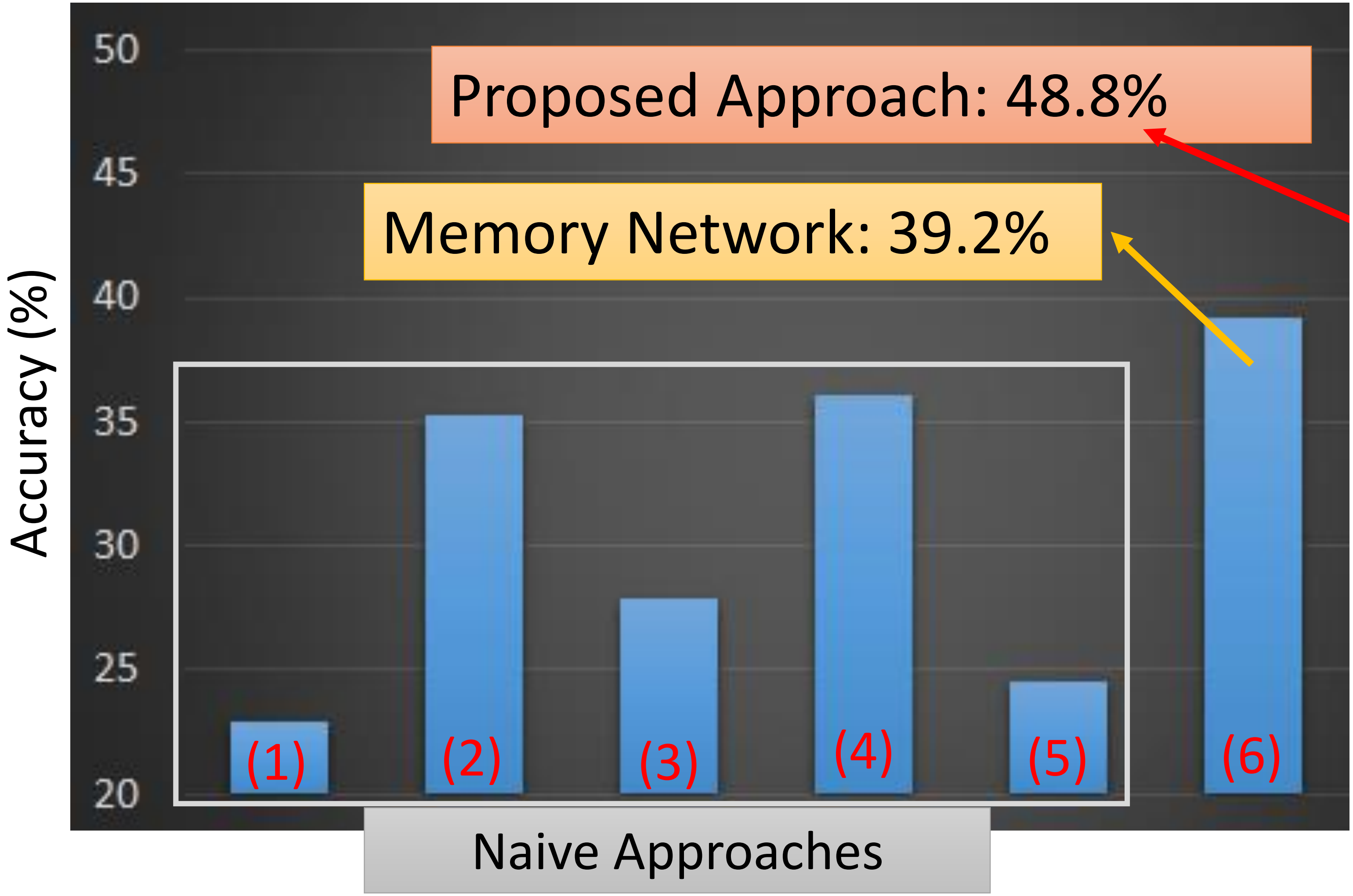


Memory Network



Proposed Approach

[Tseng & Lee, Interspeech 16]
[Fang & Hsu & Lee, SLT 16]



Analysis

Type 1: Comprehension
Type 2: Pragmatic understanding
Type 3: Connecting information, making inference and drawing conclusions